Title
On-Chip Particle Trapping and Manipulation

Permalink
https://escholarship.org/uc/item/35m9317s

Author
Leake, Kaelyn Danielle

Publication Date
2015

Peer reviewed|Thesis/dissertation
9 Conclusion ........................................................................................................... 96

10 Appendices ........................................................................................................... 99

10.1 Simulation Software ..................................................................................... 99

10.2 Particle Tracking Software ......................................................................... 133

11 Bibliography ...................................................................................................... 160
List of Figures

Figure 2-1 Schematic of ARROW layers with alternating layers with indices of refraction $n_1$, $n_2$, $n_s$, and $n_c$ where $n_1$, $n_2 > n_c$ .......................................................... 4

Figure 2-2 Schematic of chip with x-polarized light split into TE and TM components ........................................................................................................................................ 6

Figure 2-3 ARROW chip layout .................................................................................................................. 6

Figure 2-4 Schematic of transmission through ARROW chip ......................................................... 7

Figure 2-5 Optical force on particle when the particle is centered and out of center ... 9

Figure 2-6 Schematic of particle reflections and refractions in bead used to calculate $Q_s$ and $Q_g$. ................................................................................................................... 10

Figure 2-7 Schematic of particle in loosely focused Gaussian beam ......................... 11

Figure 2-8 Plots of $F_g$ and $F_s$ vs. (a) radius and (b) and (c) axial distance and radial offset ........................................................................................................................... 12

Figure 2-9 (a) Single and (b) dual beam traps ............................................................ 12

Figure 2-10 Schematic of ARROW chip used for pushing, trapping, and detection.. 13

Figure 2-11 (a) Still image of the particle being pushed along the channel and laterally centered due to the gradient force. (b) Plot of experimental data, fit used to calculate loss and initial velocity. (c) Calculated mode and measured particle trajectory of multimode liquid core. (d) Multi-mode liquid core waveguide position histogram (top panel), calculated mode (line) and averaged, calculated intensity potential (circles). d) Single-mode liquid core histogram (top panel), calculated mode (line) and averaged, calculated intensity potential (circles) [51]. ......................... 15

v
Figure 2-12 (a) Measured particle trajectory in liquid core and histogram of particle’s transverse (x) position, showing tightly centered particle motion in presence of optical guiding beam. Fluorescence collected from liquid core (b) without and (c) with guiding beam. While a single beam manipulating particles inside a liquid core waveguide adds a large number of capabilities to a planar optofluidic platform, it is the use of multiple beams that enables additional particle manipulation options, in particular particle trapping [52].

Figure 2-13 (a) Schematic of forces and loss in liquid core waveguide. (b) Trajectory of particle in trap. Particle enters from the left [40].

Figure 2-14 (a) Schematic of forces and loss in liquid core waveguide. (b) Photon counts of trapped particles, initially one particle is trapped, particles concentrate to create larger signal. c) Image showing two simultaneously trapped particles: one particle is trapped with a loss-based trap (horizontal) and the second is trapped with a divergence-based dual beam trap (vertical) [41].

Figure 2-15 a) Schematic of on-chip implementation of ABEL trap. b) Time versus fluorescence and z position showing photobleaching of trapped E. coli, fluorescence single fitted with a monoexponential decay curve.

Figure 2-16 Molecule with Raman scattering due to oscillations.

Figure 2-17 Energy Diagram of Scattering.

Figure 2-18 MMI indices and Modes.

Figure 2-19 75μm wide x 1.9mm long MMI with input wavelength 740nm and a 5μm input waveguide. This results in an output of 6 self-images.
Figure 3-1 SEM image of solid/liquid-core ARROW waveguide with (a) 3μm top oxide and d_τ=1.84μm and (b) 6.23μm top oxide and d_τ=3.76μm. [72] ...................... 31
Figure 3-2 Fundamental mode coupling efficiencies [72]............................................. 32
Figure 3-3 Simulations showing interface transmission [72] ................................. 34
Figure 3-4 SEM of (a) traditional and (b) pedestal Ta_2O_5 samples [78]. ............... 35
Figure 3-5 PL of SiO_2, SiN and Ta_2O_5 (pre and post annealing) films [78].............. 38
Figure 3-6 (a) Background fluorescence signals from SiN/SiO_2 and Ta_2O_5/SiO_2 samples. Laser is on from 10s to 40s. (b) Detection of 100nm tetraspeck nanoparticles in a SiN/SiO_2 sample (c) and on a Ta_2O_5/SiO_2 sample [78]. ......................... 40
Figure 4-1 a) Total force magnitude (dark red for strong forces, dark blue for no force) b) Quiver plot showing velocity direction and magnitude. .............................. 43
Figure 4-2 Trajectory of bead in loss based trap. Two equal forces causing particle to trap in center of channel .............................................................. 46
Figure 4-3 Trajectory of a 0.5μm particle with Brownian motion.............................. 48
Figure 4-4 Flow profile in a straight liquid core ....................................................... 49
Figure 4-5 Intersection flow profile at an intersection with 10μm/s upwards and 40μm/s to the left ................................................................. 50
Figure 4-6 Combined flow and optical force simulation showing bead velocity field and simulated trajectory. Flow as in Figure 4-5 with 30mW laser acting upwards.... 52
Figure 4-7 a) Frame of interest with 0.5μm particle b) background frame c) Frame of interest with background subtracted d) sections of image that are not the waveguide
are removed e) Converted to black and white f) small white pixel sections removed
..................................................................................................................................... 54

Figure 4-8  a) Trajectory and b) flow field profile of intersection with no flow into the upper channel and no laser.......................................................................................................................... 57

Figure 5-1 (a) Force diagram of SBT, (b) DBT, and (c) OBT. (d) Plot showing trapping point at  \( z_f \) for fixed  \( x_f \) location. ........................................................................................................ 60

Figure 5-2 Analytically calculated forces. (a) Scattering force of Beam 1 along x and gradient force of Beam 2 along z. The arrows mark the range in which a trapping point must occur and are used to create the gradient force and scattering force maximums in (b). ........................................................................................................ 63

Figure 5-3 Results from dynamic simulations with no diffusion (a) trajectory of bead (Beam 1 is red and Beam 2 is blue) (b) total force at intersection and (c) particle velocities along x and z.............................................................................................. 64

Figure 5-4 (a) potential profile and (b) trap stability. .............................................. 65

Figure 5-5 Trajectory of 3μm bead with 5μm beam waist which is unable to trap due to having a diameter greater than 2.4 times the beam waist. .............................. 67

Figure 6-1 (a) Photo of actual and (b) simplified schematic of “H” mask chip ........ 69

Figure 6-2 Instantaneous velocity vectors from optical forces of a (a) 0.25μm and (b) 1.5μm bead................................................................................................................................. 70

Figure 6-3 Several flow trajectories simulated with Brownian motion............... 71

Figure 6-4 Layout of orthogonal sorting technique............................................... 73

Figure 6-5 Flow profile of chip.................................................................................. 74
Figure 6-6 Orthogonal Orientation (0.25μm is gold, 0.5μm is red, 1μm is green and 1.5μm is blue) (a) Velocity vector field with flow and laser. (b) Flow trajectories comparing experimental (dots) and calculated (lines). (c) Simulated trajectories with $v_z=10μm/s$, $v_y=0μm/s$ and $P=20.1mW$. (d) Sorting efficiencies for $v_z/v_x=0.3$.

Figure 6-7 Layout of counterpropagating orientation

Figure 6-8 Flow profile counterpropagating orientation

Figure 6-9 Counterpropagating orientation (0.25μm is gold, 0.5μm is red, 1μm is green and 1.5μm is blue) (a) Velocity vector field with flow and laser. (b) Flow trajectories comparing experimental (dots) and calculated (lines). (c) Simulated trajectories with $v_z=v_x=10μm/s$ and $P=25mW$. (d) Sorting efficiencies for $v_z/v_x=0.7$.

Figure 6-10 Simulated particle trajectories for a range of indices for a 3μm particle with $v_z=20m/s$ and $P=30mW$

Figure 7-1 (a) Schematic of MMI chip and (b) SEM of chip

Figure 7-2 Simulated MMI spot pattern for a range of wavelengths

Figure 7-3 (A) DyLight830 absorption/emission spectrum. (B) Fluorescence at 876nm and at (C) 753nm

Figure 7-4 Simulated (lines) and measured (dots) spectra for 876nm and 753nm

Figure 7-5 Measured and simulated spectrum for 878nm with and without back reflections

Figure 7-6 Multiple particles trapped

Figure 7-7 Interwoven spot patterns of 876nm (orange) and 753nm (red)
Figure 7-8 Conveyor belt screen shots (876nm is orange and 753nm is red)............. 86
Figure 7-9 Trapped Particles with (a) 876nm and (b) 753nm. (c) Trajectory of first bead trapped with 876nm and (d) potential of same bead. ..................................................... 87
Figure 8-1 A 2μm orange fluorescent carboxolate-modified polystyrene beads with silver nanoparticle coating ................................................................. 89
Figure 8-2 SERS signal from 12.5M R6G solution with coated and uncoated beads 90
Figure 8-3 UV/Vis extinction of Ag coated microparticles................................. 91
Figure 8-4 a) Particles in groups of 1, 2, and 3. b) R6G intensity is shown relative to the number of beads in a grouping.Inset is taken from height of the peak marked with an arrow. .......................................................................................................................... 92
Figure 8-5 Microparticle with NanoFrames ....................................................... 93
Figure 8-6 (a) Setup used to get SERS spectrum on ARROW (b) SERS spectrum from ARROW waveguide................................................................. 94
Figure 8-7 Schematic and overhead image of particle in trap ....................... 95
List of Tables

Table 3-1 Measured top oxide and crevice thicknesses .............................................. 32
Table 3-2 Normalized throughput of samples [72] ..................................................... 33
Table 7-1 Stiffness of Traps for 876nm ..................................................................... 86
Abstract

Kaelyn Danielle Leake

On-Chip Particle Trapping and Manipulation

The ability to control and manipulate the world around us is human nature. Humans and our ancestors have used tools for millions of years. Only in recent years have we been able to control objects at the micron level. In order to understand the world around us it is frequently necessary to interact with the biological world. Optical trapping and manipulation offer a non-invasive way to move, sort and interact with particles and cells to see how they react to the world around them. Optical tweezers are ideal in their abilities but they require large, non-portable, and expensive setups limiting how and where we can use them.

In order to use optical manipulation in more applications and situations there is a need to integrate these techniques with lab-on-a-chip devices. We focused on the Liquid-core Anti-Resonant Reflecting Optical Waveguide (liquid-core ARROW) for our work. The ARROW chip is an ideal platform for this, it has anti-resonant layers which allow light to be guided in liquids, allowing for particles to easily be manipulated. It is manufactured using standard silicon manufacturing techniques making it easy to produce. The planar design enables integration with other technologies making it possible to combine with existing lab-on-a-chip technology.

Here, I present improvements of the ARROW chip performance by reducing the intersection losses and thus increasing transmission by a factor of 17.1 and by reducing the fluorescence and background on the ARROW chip by a factor of 10.
The ARROW chip has already been used to trap and push particles along its channel but here I introduce several new methods of particle trapping and manipulation on the ARROW chip. Traditional two beam traps use two counter-propagating beams. A trapping scheme that uses two orthogonal beams which allow for trapping at their intersection is introduced. This scheme is analyzed using realistic conditions. Simulations of this method were done using a program which looks at both the fluidics and optical sources to model complex situations. These simulations were also used to model a new sorting method which combines fluid flow with a single optical source to automatically sort dielectric particles by size in waveguide networks. These simulations were shown to be accurate when implemented on-chip. Lastly, I introduce a particle trapping technique that uses Multimode Interference (MMI) patterns in order to trap multiple particles at once. The location of the traps can be adjusted as can the number of trapping location by changing the input wavelength. By changing the wavelength back and forth between two values this MMI can be used to pass a particle down the channel like a conveyor belt.
Acknowledgements

I would like to thank Dr. Holger Schmidt, without his guidance this dissertation would not be possible. He offered me advice and direction which were frequently necessary to get me through the challenges of the degree.

I would also like to thank my undergrad institution Sweet Briar College. Though the school may be closing, it will always be in the hearts of the alumni and faculty. I particularly want to thank the engineering and physics professors: Hank Yochum, Scott Pierce, Timothy C. Scott, Scott Hyman and Dorsa Sanadgol for giving me a good education, hands on experience and the confidence to continue.

I would like to thank my colleagues, who not only helped with my research but also my peace of mind: Dr. Tom Yuzvinsky, Dr. Ashawaraya Shalini, Damla Ozcelik, Josh Parks, Jennifer Black, Yu Yahagi, Anik Duttaroy, Dr. Rebekah Brandt, Dr. Philip Measor, Dr Shuo Liu, Dr. Cathy Hong Cai, Cassidy Berk, Soren Almquist, Dr. Aiqing Chen, Dr. Bin Wu, Dr. Mikhail Rudenko, Dr. Zhigang Liu, Kavit Kumar, Melissa Eberle, Samantha Downey, Shamik Mascharak, Tamara Pico, Jinbo Shu, Mike Jaris, Gopikrishnan Gopalakrishnana Meena, Alexandra Stambaugh, Md. Mahmudur Rahman, Jason McDowell, Mark Oehlberg, Gregory Jorgensen, and Filipe Neira.

The faculty and staff at UCSC have both been helpful academically and with research and equipment needs. None of this would be possible without the support of the W.M. Keck Foundation, the National Institute of Health, the National Science Foundation and the W.M. Keck Center for Nanoscale Optofluidics.
I would like to thank my family for putting up with many years of academic struggles. My mom and dad, Kathy and Dan, offered mental and monetary assistance, and years of help. I can’t stress how important my parents have been. My sister Juliana, is both my friend and my family, she always seemed to outshine me academically and never let me stop trying to improve. Most of the time the competition was one sided but it was extremely important to my success. I would also like to thank my Aunt Linda and Uncle Tom, they are as close to parents as you can get when you already have both. I would also like to thank Ralph Parker who spent years teaching me to read, without the written word I wouldn’t have made it through high school let alone graduate school. Reading is a big part of my life and I’m thankful I got the support I needed.
1 Introduction

Since the dawn of tools and technology society has longed for faster, cheaper, smaller and more effective technology. Things once made by skilled craftsmen with limited time can now be made by automated robotics. Computers once took up rooms and now can be as small as a few inches. Improvements in silicon manufacturing techniques have allowed for small circuits and fast computers, at ever lower prices. These manufacturing techniques have led to improvements in our capabilities to produce not only semiconductors but also to miniaturize other technology such as motors [1], steam engines [2], gyroscopes [3], and deformable mirrors [4]. These miniature versions allow for small embedded systems in devices such as cars and cell phones. Unfortunately, there are still many systems that could benefit from the size and cost reductions.

Even with the immense improvements in the understanding of medicine and health, the majority of medical testing still needs to be performed by skilled technicians in specialty labs. This means tests take time, and portability and cost are limiting factors. One answer to this problem that has been rising in importance over the last decade is lab-on-a-chip platforms. The platforms already have been used for testing new drugs [5], detecting cardiac biomarkers [6], measuring lactose and glucose levels [7], and DNA sequencing [8]. Infectious diseases cause a large portion of deaths, particularly in underdeveloped countries [9] where large and expensive
equipment isn’t affordable. Methods for early detection of these diseases could save many lives and potential prevent large outbreaks.

In order to duplicate multi-step lab procedures microfluidic control, particle control, and measurement or detection are required [10]–[12]. Procedures such as pumping [13], mixing [14], [15], sorting [16]–[19], and trapping [20]–[23] are all necessary in many applications. Though some techniques exist for these, they are limited in what they can do and in what applications they will work. Current techniques for sorting, trapping and manipulation include optical [24]–[26], magnetic [27], [28], electrical [18], [29], acoustic [30], [31] and mechanical [16], [32], [33] methods. Some techniques require aspects of detection beforehand [27], [34] whereas others are automated. Optical sorting, manipulation and trapping techniques have been developed using many optical structures and techniques including solid-core waveguides [19], 1D photonic crystal waveguides [35], microrings [36], [37], slot waveguides [38], photonic crystal fibers [39], and liquid-core waveguides [21], [40], [41].

In this thesis, I introduce several manipulation techniques particularly well suited to on-chip situations. These techniques are ideal for the ARROW liquid core platform [42]–[44] which I use throughout this document. I introduce a simulation procedure to create accurate trajectories and determine the forces and velocities of particles within the channel with simple and complex flow profiles. These simulated trajectories can then be compared with particle trajectories taken from video screenshots.
In the second chapter of this thesis I introduce the reader to some key topics by explaining necessary background information. The reader is introduced to ARROW waveguides, optical manipulation on and off chip, Raman and SERS (surface enhanced Raman scattering) and the MMI (multimode interference) waveguide. In chapter 3, I discuss two platform improvements, the reduction on loss at the intersection and the reduction of fluorescence on chip. In the fourth chapter, I explain the particle tracking and simulation methods I developed for my trapping and manipulation work. This is followed by a section about a orthogonal trapping technique which uses two orthogonal beams to trap a particle. In chapter 6, I discuss an automatic sorting technique that uses a combination of fluid flow and optical power to sort particles by size. This method features two orientations both discussed in this section, one where the flow is orthogonal to the laser propagation and one in which the flow is acting in the opposite direction of the laser propagation. Chapter seven is home to an explanation of a tunable multi-spot trapping technique, which uses a MMI waveguide. This technique can be used to trap multiple particles simultaneously or to control the movement of one or multiple particles along the channel. In chapter 8, a method of on-chip SERS trapping is discussed which uses SERS active beads to detect the analyte in the waveguide. Finally a summary and outlook will be given in the conclusion in chapter 9.
2 Background

In this section I introduce the reader to the background necessary for understanding my thesis. First the antiresonant reflecting optical waveguide (ARROW) is introduced and explained. Then in the next two sections I discuss optical particle manipulation and how it’s performed on-chip. This is followed by sections introducing Raman scattering and the multi-mode interference (MMI) waveguides.

2.1 ARROW Waveguides

The antiresonant reflecting optical waveguide (ARROW) is designed to guide light in a low index core $n_c$. This is done by using alternating layers of higher index materials $n_1, n_2 > n_c$.

These alternating layers start at a silicon substrate, $n_s$. Light incident on the surface of the core is partially reflected and partially refracted, since the conditions for total internal reflection (TIR) aren’t satisfied (see Figure 2-1). [45] In order to guide in the core, the antiresonant condition must be satisfied, $\Phi_{RT} = M \pi \ (M = 1,3,5...)$, or a
round trip change in phase, $\Phi_{RT}$, that results in light remaining in the core. This can be rewritten in terms of wavelength,

$$t_i = \frac{M \lambda_D}{4n_i \sqrt{1 - \frac{n_i^2}{n_c^2} + \frac{\lambda_D^2}{4d_c^2 n_c^2}}}$$  \hspace{1cm} (2.1.1)

where $n_i$ is the index of the $i_{th}$ layer, $\lambda_D$ is the design wavelength, $d_c$ is the thickness of the core [45], [46]. By adding layers you can decrease the loss of the waveguide. The minimum loss can be defined for the transverse electric (TE) and transverse magnetic (TM) modes. The minimum TE loss is

$$\alpha_{TE} = \left( \frac{n_2^2 - n_c^2}{n_1^2 - n_c^2} \right)^{N/2} \alpha_0,$$  \hspace{1cm} (2.1.2)

where $N$ is the number of layers, $m = 1, 2, 3...$ is the mode order, materials $n_2$ and $n_1$ are alternating layers, $n_2$ is the outer layer, and $n_c < n_2 < n_1$ and

$$\alpha_0 = \frac{\lambda_D^2 (m + 1)^2}{n_c d_c^3 \sqrt{n_1^2 - n_c^2}}.$$  \hspace{1cm} (2.1.3)

The minimum TM loss is

$$\alpha_{TM} = \left( \frac{n_1}{n_2} \right)^{2N} \frac{n_1^2}{n_c^2} \alpha_{TE}.$$  \hspace{1cm} (2.1.4)
The loss in the waveguide varies with both the layers and the core index, making guiding different for water and other liquids [46], [47]. For an ARROW with x-polarized light,

\[ \alpha_{TE} \]

the vertical guiding is TE and thus the loss in that direction can be found using \( \alpha_{TE} \), conversely the horizontal loss comes from the TM direction. The opposite scheme holds true with y-polarized light. To find the total loss of the waveguide the TE and TM losses can be added (see Figure 2-2).

Figure 2-2 Schematic of chip with x-polarized light split into TE and TM components

Figure 2-3 ARROW chip layout
When working with solid-core ARROWS and liquid-core ARROWS there is also an additional loss at the intersections between the types of waveguides and that of coupling in and out of the device. This makes the total transmission of the system, 

\[ T = \kappa_{ei} e^{-\alpha_{SC} l_{sc}} \cdot \kappa_i^2 e^{-\alpha_{LC} l_{lc}} \cdot \kappa_{e2}, \]  

(2.1.5)

where, \( \kappa_{ei} \) and \( \kappa_{e2} \) are input and output coupling losses, \( l_{SC} \) and \( l_{LC} \) are the lengths of the liquid and solid cores and \( \kappa_i \) is the decrease in transmission due to the intersections (see Figure 2-4). This intersection loss is mainly due to scattering and is a limiting factor in the reduction of loss in the channel.

![Figure 2-4 Schematic of transmission through ARROW chip](image)

### 2.2 Optical Particle Manipulation

Though massless, photons have momentum and are therefore capable of transferring this momentum to the objects they collide with. The momentum transferred is small and without the consistency of coherent light other effects overpower this radiation force. The introduction of the laser allowed for particles to be pushed in the direction of propagation. Given certain conditions particles can also be trapped, allowing for the study of cells, small forces and bonds, and the tracking of movement. [20], [22], [48], [49]
A photon’s energy is related to its momentum, $\frac{h}{\lambda}$, and depends only on its wavelength, $E = \frac{hc}{\lambda}$, where $h$ is Planck’s constant, $\lambda$ is the wavelength of the light, and $c$ is the speed of light. The light hits the particle and some light is reflected, some is refracted, and a portion is absorbed (see Figure 2-6). The light that makes it through both interfaces exits the particle at a different angle due to Snell’s law, $n_1 \sin \theta_1 = n_2 \sin \theta_2$, and this means that light itself experiences a change in momentum, this momentum change creates an equal and opposite force on the particle with a magnitude of $F = \frac{\Delta p}{\Delta t}$. [23]

If the particle is located in the center of a collimated beam the particle will be pushed along the direction of the beam, since the momentum change is symmetric with respect to the particle’s center. If the particle is off center, the sum of the momentum change will push the particle in the direction of the beam, and due to the differences in power in the center and edges of a Gaussian beam, along the gradient. This is due to a relatively larger number of photons in the center of the beam then at the outside of the beam (see Figure 2-5). The component of the optical force in the direction of propagation is typically referred to as the scattering force, and the component of the force acting along the intensity gradient is known as the gradient force. The gradient force will act to “draw” a particle into the area with the highest intensity, if the particle’s index is larger than the medium it moves in; this means that if a beam is focused there is also a gradient force that pulls the bead to the focus point.
The force imparted by a single photon is very small; the combined force of all the photons in a beam is what we observe. In general the force of a single ray of light per second can be described as

$$F = Q \frac{n_i P^2}{c}$$  \hspace{1cm} (2.2.1)

where $Q$ is a dimensionless constant that defines the interaction of the light with the particle, $n_i$ is the index of the medium in which the particle resides, and $P$ is the power of the ray. The value of $Q$ depends on the component of the optical force, $Q_s$ for the scattering force, $Q_g$ for the gradient force, and for the magnitude of the total force, $Q_{mag} = \sqrt{Q_s^2 + Q_g^2}$. The value for the scattering force is,

$$Q_s = 1 + R \cos(2\theta) - \frac{T^2 \left[ \cos(2\theta - 2\theta) + R \cos(2\theta) \right]}{1 + R^2 + 2R \cos(2\theta)}.$$  \hspace{1cm} (2.2.2)
For the gradient force,

\[
Q_g = R \sin(2\theta) - \frac{T^2 \left[ \sin(2\theta - 2r) + R \sin(2\theta) \right]}{1 + R^2 + 2R \cos(2r)}.
\]  

(2.2.3)

Where \( R \) and \( T \) are Fresnel reflection and transmission coefficients, \( \theta \) is the angle of incidence, and \( r \) is the angle of refraction (see Figure 2-6). [49]

![Figure 2-6 Schematic of particle reflections and refractions in bead used to calculate \( Q_s \) and \( Q_g \).](image)

To get the value for the whole beam it’s necessary to integrate over the surface area of the sphere that interacts with the light. For particles in the ray optics regime, or \( r_p > \lambda \), in a loosely focused beam these forces can be written as,

\[
F_s = \frac{n_e}{2c} \int_0^{\pi/2} \int_0^{\pi/2} I(\rho, z) Q_s r_p^2 \sin(2\theta) d\theta d\phi
\]  

(2.2.4)

and

\[
F_g = -\frac{n_e}{2c} \int_0^{\pi/2} \int_0^{\pi/2} I(\rho, z) Q_g r_p^2 \sin(2\theta) \cos \phi d\theta d\phi
\]  

(2.2.5)
where \( I(\rho, z) \) is the intensity, \( r_p \) is the radius of the sphere, \( \phi \) is the polar angle of the bead in the beam, \( z \) is the bead's position along the beam, \( a \) is the particle's position relative to the beam center and \( \rho \) is the position of the current ray (see Figure 2-7).

![Figure 2-7 Schematic of particle in loosely focused Gaussian beam](image)

The size of the force is therefore dependent on the index of the liquid, the index of the particle, the particle's size (see Figure 2-8a), the beam's size and level of focus and the position of the particle in the beam (see Figure 2-8b and Figure 2-8c). When the particle is offset from the center of the Gaussian (\( a > 0 \)) the particle will feel a force that pulls it towards the center (\( F_g \)). This force is sinusoidal with its zero located at the center of the beam. The scattering force (\( F_s \)), on the other hand, is largest at the center of the beam. [50]
The first person to successfully observe these forces on a micron-sized bead was Ashkin in 1970 [22]. He was able to trap a bead in a single highly focused beam (see Figure 2-9a) in what we now call “Optical Tweezers” [48], this method typically uses an inverted microscope and something to steer the beam, such as an adjustable mirror. In this case the gradient force toward the focal point is used to keep the particle stable in all directions [48]. Less commonly used also introduced by Ashkin is the dual beam trap, which uses to counterpropagating beams to balance scattering forces as seen in Figure 2-9b [20].
2.3 Optical Manipulation on ARROW chip

The majority of my work was done using the ARROW chip. The following section introduces the earlier manipulation and trapping work done on this platform. This work is instrumental in the understanding of my contributions.

In 2008 it was shown by Measor et al. that particles could be pushed along an liquid-core ARROW by optical forces [51]. The ARROW chip used has a liquid core –ARROW that runs perpendicular to an solid-core ARROW (See Figure 2-10).

A particle in the channel experiences three types of forces. The scattering force, gradient force described earlier and the Stokes drag force resulting from pressure-based or electrokinetic fluid flow. The main source of the forward movement of the particle is due to the scattering force which is acting along the direction of light propagation, \( z \). In addition, transverse gradient forces exist that pull the particle towards the high-intensity regions of the mode. The scattering force is highly dependent on the amount of power acting on the particle. Due to loss, \( \alpha \), along a waveguide the power entering the waveguide \( P_0 \) decays exponentially with distance [51]. This leads to a scattering force equation that is slightly different than the
version mentioned earlier in Equation (2.2.1) since it varies with propagation distance,

\[ F_z = \frac{Q}{c} \frac{\eta}{r} P \exp(-xz) \]  

(2.3.1)

In steady state, the Stokes drag force must counteract and balance the scattering force in order to prevent acceleration,

\[ F_{\text{Stokes}} (z) = -6\pi\eta' r \frac{dz}{dt} = -F_z(z) \]  

(2.3.2)

where Therefore, the particle trajectory along the channel can be calculated to

\[ z(t) = \frac{1}{\alpha} \ln \left[ v_0 \alpha t + \exp(\alpha z_0) \right] \]  

(2.3.3)

where \( z_0 \) is the initial particle position, and \( v_0 \) is the velocity of the particle with the initial power,

\[ v_0 = \frac{Q P_0 \eta}{6\pi \eta' r c} \]  

(2.3.4)

The gradient force acts to center the particle in the channel since the direction of the gradient force, \( F_G \), is towards the highest intensity. The motion of the particles diffusion perpendicular to the propagation direction is defined by the corresponding potential \( U(x) \),

\[ U(x) = -kT \ln p(x) = -\frac{\gamma}{ce} I(x) = -\int F_s(x) dx \]  

(2.3.5)
where $p(x)$ is the probability of finding the particle at a given location, $k$ is the Boltzmann constant, $T$ is the temperature, $\varepsilon$ is permittivity, $\gamma$ is the effective polarizability of the particle, and $I(x)$ is the intensity profile of the guided mode.

The dependence of the scattering force on the particle’s position in the channel (2.3.1) enables the characterization of loss and mode profile of a hollow-core waveguide as shown by Measor et al. [51]. The particle trajectory along the channel $z(t)$ was recorded using a CCD camera in top-down view. Equation (2.3.3) can then be fitted to the measured trajectory to extract the waveguide loss as seen in Figure 2-11a.

Similarly, the particle transverse location of the particle in the channel $x(z,t)$ gives information about how the intensity of light changes along that axis. Since the gradient force pulls particles into high intensity regions, the likelihood of finding a particle in such a region is higher.

Figure 2-11 (a) Still image of the particle being pushed along the channel and laterally centered due to the gradient force. (b) Plot of experimental data, fit used to calculate loss and initial velocity. (c) Calculated mode and measured particle trajectory of multimode liquid core. (d) Multi-mode liquid core waveguide position histogram (top panel), calculated mode (line) and averaged, calculated intensity potential (circles). (d) Single-mode liquid core histogram (top panel), calculated mode (line) and averaged, calculated intensity potential (circles) [51].
A histogram of lateral position distributions was created from the movie of the particle moving along the channel under the influence of a single-beam optical force (Figure 2-11b and Figure 2-11c). The figure shows excellent agreement between the shape of these histograms and the waveguide mode profiles calculated with commercial photonic simulation software.

Measor et al. introduced another use for a single optical beam in 2009 [52]. One of the main issues with particle sensing in liquid-core waveguides is the fact that particles can assume lateral positions in the channel that result in poor coupling of any generated fluorescence into the liquid core waveguide modes. In a typical fluorescence experiment as shown in Figure 2-12a, excitation light is coupled into a solid-core ARROW with a single-mode fiber, particles at the intersection of the solid-core ARROW and liquid-core ARROW are then excited. Fluorescence from these particles is then collected along the flow channel and detected using a sensitive photodetector. By adding a near infrared guiding beam along the liquid channel, particles are pulled into the center of the channel by the gradient force. This “optical focusing” results in higher and more uniform coupling of fluorescence into the liquid-core ARROW modes, and thus in a large improvement in collection efficiency by up to 85% (See Figure 2-12b).
In 2009 Kühn et al introduced a new type of dual-beam trap based on counter-propagating beams [40]. Unlike Ashkin’s original dual-beam trap which relied on spatial variations of the beam area, this method used the propagation loss in confined liquid-core ARROW modes create beam-dependent force profiles and trap particles (See Figure 2-13a). This loss-based (LB) trap is a long-range trap in which the particles can be held at any point along a several mm long channel by adjusting the relative power of the two beams. In this case, the scattering force takes on a similar form to Eq. (2),

\[
F_{r/l}^{Scal} = \mp Q \frac{C}{\eta} P_{0/l}^n \exp \left( -\alpha \left( \frac{L}{2} \pm z \right) \right), \tag{2.3.6}
\]

where \(L\) is the length of the liquid-core ARROW, \(P_{0/l}^n\) is the input power at each end of the liquid core waveguide, and \(z\) is the particle’s position relative to the center of the liquid core. The trapping point occurs where the scattering forces are balanced:
For a waveguide with $\alpha = 5.2 \text{cm}^{-1}$ the forces were found to be about 0.14pN at the beginning of the liquid core and the trap was found to have a stiffness of about 50 nN/m (See Figure 2-13b).

$$z_1 = \frac{1}{2\alpha} \log \left( \frac{p_0}{p_t^0} \right)$$

(2.3.7)

Figure 2-13 (a) Schematic of forces and loss in liquid core waveguide. (b) Trajectory of particle in trap.

Particle enters from the left [40].

With this method Kühn et al were able to trap single silica beads as well as silica beads tagged with an Escherichia coli (E.coli) bacterium whose DNA was stained with Acridine Orange intercalating dye. This LB trapping method can be extended to multiple particles to function as a particle concentrator for increased detection efficiencies [41]. When trying to detect particles at low concentrations, near the single particle level, it can be difficult to detect the fluorescence. Here, particles from either end of the channel are drawn to the trapping point under the influence of the two trapping beams (see Figure 2-14a) This can increase the concentration of particles by more than 2 orders of magnitude (see Figure 2-14b). Once trapped the group of particles can easily be moved by adjusting relative input powers.
Moreover, a traditional divergence based (DB) trap can be created by matching beam powers via two solid-core ARROWS. Once the beam exits the solid core it diverges as it propagates across the liquid core. Due to the chip architecture this DB trap can be used simultaneously with the LB trap as shown in Figure 2-14c [40].

It was shown that this “dual dual-beam trap” can be used to control the relative spacing of two particles trapped in each beam pair. A separation of 7.4μm of each other without losing the two separate traps. Such a configuration could be used to facilitate controlled interactions and binding between two molecules tethered to trapped microbeads.

The major drawback of all-optical traps, including the chip implementations described thus far, is the relatively high laser power that is needed to generate a sufficiently deep trapping potential. One method of reducing the amount of power is to use electro-optical traps that combine an ultralow power optical source with electrical feedback. The optical source can be used for both generation of the
feedback signal required for trapping and fluorescence or scattering analysis of the trapped particle. Such traps are termed Anti-Brownian Electrokinetic (ABEL) traps and exist in various bulk optical implementations [53], [54]. The ABEL trap scales favorably with reduced particle size, and objects on the nanometer scale down to single dye molecules have been trapped in bulk versions [55]. Kühn et al demonstrated an optofluidically integrated version of an ABEL trap as shown in Figure 2-15 [21]. This trap has two solid-core waveguides that are offset from each other such that they overlap by half their mode’s beam waist. These waveguides are alternately excited, creating a modulated fluorescence or scattering signal from a particle in the intersection. By correlating the fluorescence signal with the excitation sequence, the location of the particle can be determined, and an electrical force is applied to push the particle towards the center of the intersection. This compensates for Brownian motion along the channel direction and was proven to be sufficient to trap micro and nanoscale beads with ultralow optical powers as low as 10µW, approximately five orders of magnitude lower than the all-optical version on a similar
Moreover, single, fluorescently stained E. coli bacteria were successfully trapped and the photobleaching process of its stained DNA was observed. In addition to the low power, another unique characteristic of ABEL traps is that they are true single particle traps as random Brownian motion can be corrected for only one particle.

2.4 Raman Scattering and SERS

The majority of scattered light (Rayleigh scattering) is the same wavelength as the light that produced the scattering. Raman Scattering is a type of inelastic scattering, which means that photons that are absorbed are re-emitted with a different energy than they entered. The difference in the wavelength of light (or energy) is the energy required to excite the molecule from its current vibrational state to a new state. [56] The light produced is coherent unlike the light produced by fluorescence. This form of scattering is dependent on the polarizability of the molecule. [57] A polarizable molecule is one that is naturally non-polar but after perturbation of the electron cloud a dipole moment can be created.

![Figure 2-16 Molecule with Raman scattering due to oscillations](image)

How easy it is to create this dipole moment defines its polarizability. [58] When a high intensity light such as that of a laser is transmitted through a sample, these
photons create an oscillating polarization in the molecules (see Figure 2-16). These oscillations produce virtual energy states. These states aren’t stable and therefore the photons are re-radiated. In a type of chain reaction, the initial oscillating polarizations can start other oscillations creating more output wavelengths than initially created.[56]

The spectral lines are specific to the molecule and can be used to identify the molecule present. The scattered light can be defined in terms of Raman shift which is typically reported in terms of wavenumber rather than wavelength, the Raman shift is defined as,

\[
\vec{\nu}_r = \frac{1}{\lambda_0} - \frac{1}{\lambda_1}
\]  

(2.4.1)

where \( \lambda_0 \) is the excitation wavelength and \( \lambda_1 \) is the wavelength of light produced during scattering.
Depending on the output energy, there are two types Raman scattering. If the scattered light has a wavelength higher than the input wavelength then it is said to be Stokes scattering. Conversely Anti-Stokes scattering is Raman scattering in which the spectral lines have a wavelength lower than the incident light. Anti-Stokes Raman is far weaker and therefore not always possible to observe. Standard Stokes scattering is created when the energy starts from its ground state and falls to its vibrational state, whereas Anti-Stokes scattering is when the energy starts and its vibrational state and returns to its ground state (see Figure 2-17). [57]

Raman scattering is very rare compared to that of Rayleigh scattering, with Raman scattering only happening 1 out of a million times. [56] The cross-section, or the effective area that defines the probability of scattering for Raman scattering is fairly low (~$10^{-30}$ cm$^2$) [59], [60] and is dependent on the molecule’s polarizability. The much more common Rayleigh scattering has a cross-section of around $10^{-28}$ cm$^2$ [59] and fluorescence a cross-section of about $10^{-16}$ cm$^2$ [61]. Since the Raman scattering values are so low it’s nearly impossible to detect without high intensity light and good filtering.

The intensity of light that is Raman scattered is,

$$I_r = \frac{\pi^2 c}{2\varepsilon_0} \vec{v}_r^4 p_0^2 \sin^2 \theta$$

(2.4.2)

where $c$ is the speed of light, $\vec{v}_r$ is the Raman shifted wavenumber, $p_0$ is the amplitude of induced electric dipole moment at this wavenumber, $\theta$ is the angle from
the axis of the induced dipole moment and $\varepsilon_0$ is the vacuum permittivity. [62] These relatively low values makes the natural level of Raman difficult to use.

Due to these low values it wasn’t until it was discovered that rough metallic structures and nanostructures were able to increase signals by values of about $10^{11}$ times, that Raman could be used in typical detection. [63], [64] This was first observed by Fleischmann in 1974 with Pyridine on a rough silver electrode and is now referred to as surfaced enhanced Raman spectroscopy (SERS). [65] SERS allows for detection of single molecules with its large enhancements making it viable for detection. The reasons for these enhancements are still not totally understood, there are two competing theories, the electromagnetic theory and the charge transfer theory.

The electromagnetic theory states that the incident light hits the metal surface and this creates surface plasmons. At the plasmon resonance frequency the particle experience a large increase in its local field, this allows for the molecules surface electrons to move freely making it easier to polarize. In order for enhancement to occur the plasmons must oscillate perpendicular to the surface plane. The scattering is a result of the plasmons traveling up the peaks. [56]

The charge transfer theory also known as the chemical enhancement explains the effects by the formation of bonds between the metal surface and the molecule. The theory explains that this bond creates a new species that is a combination of the molecule and the metal allowing for charges to move freely between the metal and the molecule. Due to the presence of the metal electrons the molecule will become more polarizable. This new species is thought to have resonances that act as a bridge
between the molecule and the metal allowing for the metal to radiate the Raman. This
is only possible with certain molecules and therefore cannot be used to explain all the
enhancement. [56]

The SERS process is most likely a combination of the two effects, though it’s
likely there is a larger electromagnetic enhancement. There is evidence of both effects
but it’s difficult to tell apart, leading to lack of agreement between researchers. Both
require metal surfaces, silver being ideal, though all metals seem to work to some
extent. Both methods also require the molecules to be close to the surface of the metal
in order to see enhancement.

2.5 MMI

When multiple waveguide modes of light interact they create patterns of interference
both constructive and deconstructive. This creates an interference pattern that varies
spatially. If a single input field is inputted in a waveguide that is able to support
multiple modes these modes will create an interference pattern called a multimode
interference (MMI) pattern. One property of this interference pattern is that the input
will be reproduced at certain intervals along the propagation direction. These
reproductions (self-images), can be in multiples and singles.
For the simple case of step index fiber with ridge index of $n_r$ and cladding index of $n_{c}$ (see Figure 2-18) the propagation constant for a given free space wavelength of $\lambda_0$, the propagation constant can be found by using the dispersion equation,

$$k_{yy}^2 + \beta_v^2 = k_0^2 n_r^2$$  \hspace{1cm} (2.4.3)

where $k_0 = \frac{2\pi}{\lambda_0}$, and $k_{yy} = \frac{(v+1)\pi}{W_e}$, giving a propagation constant of [66],

$$\beta_v = k_0 n_r - \frac{\lambda_0 (v+1)^2 \pi}{4n_r W_e^2}$$  \hspace{1cm} (2.4.4)
where \( k_v = \frac{2\pi}{\lambda_0} \) and \( v \) is the current mode number for a waveguide that supports \( n \) modes. \( W_v \) is the effective width of the fundamental mode,

\[
W_v = W_{w} + \frac{\lambda_0}{\pi} \left( \frac{n_r}{n_g} \right)^2 \left( \frac{n_r^2}{n_g^2} - 1 \right)^{1/2}
\]

(2.4.5)

where \( W_w \) is the width of the waveguide and \( \sigma = 0 \) for TE and \( \sigma = 1 \) for TM modes [67].

Since each mode has its own propagation constant the different modes will have different maximum amplitudes at different locations causing them to interfere. The propagation constant spacing is,

\[
\beta_0 - \beta_v = \frac{1\left( v + 2 \right)\pi}{3L_x}
\]

(2.4.6)

where \( L_x \), the beat length of the fundamental mode and the first mode, is

\[
L_x = \frac{\pi}{\beta_0 - \beta_v} = \frac{4n_rW_e^2}{3\lambda_0}.
\]

(2.4.7)

If we assume the input mode profile, \( \psi(y,0) \), is only made up of guidable modes then the profile in the MMI at any given location \( L \) can be described by,

\[
\Psi(y,L) = \Psi(y,0) \exp\left( iL(\beta_0 - \beta_v) \right).
\]

(2.4.8)
Therefore an exact self-image and a mirrored self-image will appear at,

\[ L = p \left( \frac{3l}{2} \right) \quad p = 0, 1, 2, \ldots \]  

(2.4.9)

where the exact images appears for even \( p \) values and mirrored images appear for odd values. [66]

In the case of multiple self-images (see Figure 2-19) the images will appear at,

\[ L = \frac{p}{N} \left( \frac{3l}{2} \right) \]  

(2.4.10)

where \( N \) is the number of self-images along \( y \). Rewriting this out explicitly,

\[ L = \frac{p}{N} \left( \frac{4nW^2}{\ell_0^3} \right) \]  

(2.4.11)

the position of the \( N \) Self-images is related to the indices of refraction the width of the
waveguide and the wavelength of the input light. For the same device, with a known perfect $N_6$ spots at wavelength $\lambda_0$, another perfect $N_1$ spots is found at,

$$\lambda_1 = \frac{N_1 \lambda_0}{N_1} \tag{2.4.12}$$

This means by tuning the wavelength the number of spots can easily be adjusted, creating a different interference pattern and a new output. [66]
3 Platform Improvements

The ARROW platform has been successfully used for many applications [68]–[70]. In order to have increased detection and manipulation capabilities reduction of the loss allows lower power and less signal. It’s also necessary to have high signal to noise and this means a reduction in the fluorescence of the chip itself.

3.1 Intersection loss

One of the main sources of optical loss in an ARROW chip is at the intersection between the liquid and solid cores. There is a large amount of scatter at these locations as the light hits upper layers and due to misalignment in the cores cause mode misalignment. Along the x-direction the mode-mismatch is limited by the capabilities of the lithography (1 μm). The y-direction has a lot more room for misalignment. One source of this difference in mode centers is due to the upper layers that confine the light in the liquid core. In the solid core these layers lie below the core, whereas in the hollow core the layers are above the core. This was partially reduced by designing structures without the top layers [71]. This reduces the amount of mode mismatch but also increases the loss in the liquid core, traditional structures have loss as low as $2 cm^{-1}$ compared to the $3.4 cm^{-1}$ of the single over coating (SOC) waveguides. The second major source of mode-mismatch is from the deposition profile. The ARROW chips we use are manufactured using plasma enhanced chemical vapor deposition (PECVD) like most other deposition techniques creates a crevice as a result of the shadowing effect and the differences in coating rates on horizontal and vertical services (see Figure 3-1). The depth of the crevices, $d_{cr}$, can
vary to some extent by adjusting the deposition conditions, but it is impossible to completely eliminate it. In order to increase the throughput and decrease the intersection loss the top oxide can be adjusted to find the ideal oxide thickness, or the oxide thickness with the least amount of intersection loss.

![Figure 3-1 SEM image of solid/liquid-core ARROW waveguide with (a) 3μm top oxide and d cx =1.84μm and (b) 6.23μm top oxide and d cx =3.76μm. [72]

To determine the ideal top oxide thickness to create the smallest possible crevice, four devices were created. Each chip had identical layers other than the top cladding which varied in thickness. They were fabricated by alternating cladding layers on a silicon substrate using PECVD. SU-8 was then used to create a sacrificial core that once etched became the hollow core. The top layers were deposited with the same technique as the bottom layers. Then the hollow core was etched with a wet etch. Finally, the excess top oxide was removed to create a single waveguide. The layers of alternation SiO₂ and SiN have the thicknesses of 270/93/93/270nm with a 5μm x 12μm core and top thicknesses starting with SiN of 127/285/142/300/123/d to, where d to is the thickness of the top oxide. After all optical tests were performed the
crevices were milled and imaged scanning electron microscope (SEM), thickness of crevices is defined as (see Table 3-1).

<table>
<thead>
<tr>
<th>Sample #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{so}$ (μm)</td>
<td>3.00</td>
<td>4.27</td>
<td>5.03</td>
<td>6.23</td>
</tr>
<tr>
<td>$d_{cr}$ (μm)</td>
<td>1.84</td>
<td>2.70</td>
<td>3.14</td>
<td>3.76</td>
</tr>
</tbody>
</table>

Table 3-1 Measured top oxide and crevice thicknesses

Figure 3-2 Fundamental mode coupling efficiencies [72]

In order to determine the loss of the ARROW waveguides the amount of light scattering was measured from the top. It’s then assumed that the amount of light being guided is proportional to the amount of light being scattered [73], [74]. Images
were taken from above using a video camera (Fastcam SA.3) and then analyzed using MATLAB. The coupling efficiency between the solid and liquid cores was determined by finding the change in scattering before and after the intersection for each of the samples (see green). The coupling efficiencies vary from 0.18% to 67% as the oxide thickness increases. The error bars in this plot show the standard deviation of different samples. The improvements in the interface transmission can also be seen by measuring the overall throughput of the waveguide. The actual transmission values of the four samples is shown in Table 3-2, a 17.1x increase in transmission over the increase in thickness was found.

<table>
<thead>
<tr>
<th>Sample #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d_{so}(\mu m))</td>
<td>3.00</td>
<td>4.27</td>
<td>5.03</td>
<td>6.23</td>
</tr>
<tr>
<td>Normalized Throughput</td>
<td>0.06</td>
<td>0.22</td>
<td>0.63</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 3-2 Normalized throughput of samples [72]

Using FIMMPROP by Photon Design, a simulation of the chip was performed to find values for the same coupling. We initially performed the simulation with no crevice present (ideal scenario, \(d_{so} = d_{cr}\)). In this case, the optimal oxide thickness was found to be \(d_{so} = 4.5\mu m\) (see blue). This value makes theoretical sense since the hollow core is 5 \(\mu m\) thick and, therefore, as the solid core height increases the center of its mode becomes higher than that of the hollow core. Since transmission values
increased across the oxide thickness range we tested, $3\,\mu m - 6.23\,\mu m$, this simulation as expected doesn’t agree with the values we got experimentally.

![Figure 3-3 Simulations showing interface transmission](image)

We performed a second simulation by using the values from Table 3-1 to create a more realistic interface profile. With these values we found that there was an increase in transmission at the intersection up to a thickness of $d_o = 7\,\mu m$. This matches the trend of the experimental values. The results of the simulation (see Figure 3-3) show a large portion of scatter for the thinner top oxide value. Overall the simulations yielded higher transmission values then our actual results, some causes for this might include reflections from upper layers on side walls, core roughness and imperfections and variations of thickness throughout the sample. We found that for oxides thicker than $6.5\,\mu m$ deposited with PECVD there was an increase in the defects and these stress lead to layer delamination.

### 3.2 Fluorescence Reduction

Detection of low concentrations of analyte is necessary particularly in cases of early detection of contagious diseases. This means that the signal to noise (SNR) level
needs to be as low as possible (SNR = average detected signal / standard deviation of the noise). Photoluminescence (PL) in the cladding layer of the waveguide can cause noise in the wavelength range of the fluorescence you are trying to measure. This can significantly increase the noise, as it cannot be filtered out.

The majority of ARROW structures have been made with silicon dioxide SiO$_2$ and silicon nitride SiN [75], [76]. SiN has a large amount of PL within the region of interest for many fluorescence detection situations making the PL add to the background noise and making it difficult to detect the analyte. One alternative material is tantalum oxide, Ta$_2$O$_5$ that has a refractive index similar to SiN and has good adhesion to SiO$_2$. It also can be deposited with sputtering, features a high melting point, low PL and is resistant to acids; all qualities that make it a good material to replace SiN [77].

![Figure 3-4 SEM of (a) traditional and (b) pedestal Ta$_2$O$_5$ samples [78]](image)

Ta$_2$O$_5$/SiO$_2$ ARROWs are made in much the same was as their more traditional counter parts [76]. Alternating layers of the two materials are deposited, the SiO$_2$ with plasma-enhanced chemical vapor deposition (PECVD) and Ta$_2$O$_5$ by
sputtering. The Ta$_2$O$_5$ can be deposited with PECVD but the process is difficult [79] and therefore the simpler process of sputtering is used. The temperature of the sputtering and the PECVD are kept about 250°C in order to ensure they are dense enough to survive the etching bath that is required for the core. Ta$_2$O$_5$ can be used to make SOC or traditional ARROW samples with or without pedestal design [80] (see Figure 3-4).

In order to compare the loss of the Ta$_2$O$_5$ samples with their counterparts, solid-core ARROW waveguides were created with layers SiO$_2$/Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$/SiO$_2$ (281/93/281/93/281/93/93-5000) where measurements are in nm and the last layer is the thick top oxide. By measuring transmission of the solid cores, reducing the length, and re-measuring transmission, the loss coefficient was found to be $0.9 \pm 0.1 \, \text{cm}^{-1}$ compared to $0.7 \pm 0.1 \, \text{cm}^{-1}$ for SiN. The liquid core samples used had core that were 12x5 $\mu$m$^2$ with layers (in nm) of SiO$_2$/Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$ - core-Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$/SiO$_2$/Ta$_2$O$_5$/SiO$_2$ (281/93/281/93/281/93-5000-158/281/158/281/158/5000). For the liquid cores loss was measured by measuring the scatter along the core and assuming this was proportional to the power within the channel [74], [81], the loss was found to be $3.1 \pm 0.2 \, \text{cm}^{-1}$ compared to $2.7 \pm 0.2 \, \text{cm}^{-1}$ for SiN. Though values are slightly higher for Ta$_2$O$_5$ they are comparable and thus the loss shouldn’t be a problem.

The initial reasoning of switching to Ta$_2$O$_5$ was to reduce PL in the waveguide in the region of interest. For a pump beam of 633nm, most detection occurs between 660 and 690nm, outside of this region filters can be used to remove
unwanted PL. In order to test the relative levels of PL 150nm films were created for both SiN and Ta$_2$O$_5$ which were deposited in the same manner as on the full ARROW chip. These samples were then tested using a Jobin Yvon LabRAM HR spectrometer ($\lambda_{\text{ex}} = 633\text{nm}$, $P=5\text{mW}$, $t_{\text{integration}} = 5\text{s}$). We measured the PL of the Ta$_2$O$_5$ before and after being annealed at 250°C since in order to grow the SiO$_2$ films it is necessary for the Ta$_2$O$_5$ to be at this temperature. The results shown in Figure 3-5 show that SiN has higher PL than both SiO$_2$ and Ta$_2$O$_5$, it also shows that the annealing process reduces the PL in the sample. One possible reasoning is that annealing in an oxygen-free chamber reduces the density of the midgap trap states that cause PL [82].
Figure 3-5 PL of SiO$_2$, SiN and Ta$_2$O$_5$ (pre and post annealing) films [78]

The complete change in noise level of the Ta$_2$O$_5$ compared to the SiN was measured by using an optical detection setup. 633nm pump light was coupled in the ARROW via the solid-cores, any fluorescence was then collected via an Olympus objective lens with a NA=0.85. To reduce the noise, fluorescence was collected perpendicular to the pump and two optical filters (640nm long pass and 670nm bandpass 17nm) were used. The signal was then detected by an avalanche photodiode (Perkin Elmer, SPCM-AQR-14-FC) and analyzed using a single photon counting card and Picoquant’s Timeharp 200 software. The total transmission for both Ta$_2$O$_5$ and SiN chips was about 10%. After filling the chips with water the number of counts was
measured with the laser off \((t < \tau_5 \text{ and } t > 4\tau_5)\) and with the laser on \((1\tau_5 < t < 4\tau_5)\). It was found the Ta\(_2\)O\(_5\) reduced the noise by a factor of 10 compared to the SiN.

The final goal was to reduce the SNR and increase the number of particles detected. In order to test this improvement, 100nm Invitrogen Tetraspeck beads were added. In order to have approximately one bead in the detection region (85 fl using an \(1/e^2\) approximation) a concentration of \((8.8\times10^{10}\) particles/ml was chosen. The particles were moved down the channel via pressure-based flow, and detected using the same setup and APD as in the background count measurement done above. Both samples were measured over a 5s span, spikes above the noise level correspond to a nanoparticle being detected see Figure 3-6c for the Ta\(_2\)O\(_5\) results and Figure 3-6c for the SiN plot. It’s easy to see simply by looking at the plot that the background is higher for the SiN than the Ta\(_2\)O\(_5\). Over a 40s time interval there was 948 particles detected and a SNR of 126.7 for the Ta\(_2\)O\(_5\) sample and only 53 particles detected and a SNR of 10.3 for the SiN sample. This is an improvement in the number of particles detected of a factor of 18 and an improvement factor of 12 times for the SNR.
Figure 3-6 (a) Background fluorescence signals from SiN/SiO₂ and Ta₂O₅/SiO₂ samples. Laser is on from 10s to 40s. (b) Detection of 100nm tetraspeck nanoparticles in a SiN/SiO₂ sample (c) and on a Ta₂O₅/SiO₂ sample [78].

Overall the Ta₂O₅ films offer a large decrease in the PL in the region of interest between 660nm and 690nm while maintaining similar loss values for both the solid and liquid-core waveguides. There was a factor of 10 improvement in the background
noise, a factor of about 12 in the SNR, and we were able to detect a significantly higher number of particles. \( \text{Ta}_2\text{O}_5 \) is overall a great replacement for the SiN previously used in ARROW waveguides.
4 Simulation and Tracking

The parameters of a trapping or sorting technique need to be determined in order to properly predict and understand what is happening. I’ve developed methods for both simulating and tracking particle trajectories using MATLAB. The simulation technique combines optical forces, fluidic forces and Brownian motion to describe the particle’s movement. The tracking program looks at recorded videos and locates particles, determines the sizes of those particles, and tracks particles as they move in a channel. Both can be used to verify trapping and sorting.

4.1 Simulation

An object in a liquid-core ARROW experiences forces from the movement of the liquid that is surrounding it and from the optical forces that interact with it. There also is some component of movement due to Brownian motion. The larger the object the more optical force, the more fluid force and the smaller the effect of Brownian motion is on its movements. In order to get an accurate picture of how all the components interact it is necessary to look at all the components individually. Then forces can be combined and velocities determined.

The optical forces in a linear waveguide can be approximated using a collimated Gaussian beam. This means that equations (2.2.4) and (2.2.5) for loosely focused and collimated Gaussian beams can be used within our waveguides [50]. These equations allow for the force on a particle to be determined based on where the particle is within the beam. The double integral can be performed numerically using MATLAB. For the
simple case of a single optical beam, with no flow, the forces on a particle can be determined by solving the above equations for the current location of the particle.

Figure 4-1 a) Total force magnitude (dark red for strong forces, dark blue for no force) b) Quiver plot showing velocity direction and magnitude.

within the channel. For multiple optical beams, the gradient and scattering force can be calculated for each beam and then combined to total forces. This technique can be performed for a particle as it moves down a channel or can be used to create a field showing the magnitude and/or direction of a force for a specified particle at any point within the channel. Graphically this allows for potential trapping spots to be seen. Plots showing the forces in the LB trap [40] mentioned earlier can be seen as a
magnitude plot in Figure 4-1a and as directional arrow plot in Figure 4-1b below, the loss is unrealistically high to make the effects obvious over a shorter distance.

These force values can be used to determine the instantaneous velocity of the particle that is experiencing these forces. At any location, a particle of a specific size will feel a scattering force and a gradient force from all beams present. This can be split into two forces, along \( z \) and \( x \).

\[
F(z) = F_{sz}(z) + F_{gz}(z)
\]
\[
F(x) = F_{sx}(x) + F_{gx}(x)
\]

where \( F_{sz} \) is the sum of the scattering forces from all the beams that propagate along the \( z \) direction and \( F_{gz} \) is the sum of the gradient forces from all the beams that propagate along the \( z \) direction. The forces that act along the \( z \) direction work the same but with opposite scattering and gradient dependence. The velocity of the particle can be determined for both these directions using Stokes’ Law,

\[
F_{stokes}(z) = -6\pi \eta' r \frac{dz}{dt} = -F(z)
\]
\[
F_{stokes}(x) = -6\pi \eta' r \frac{dx}{dt} = -F(x)
\]

where \( r \) is the radius of the particle and \( \eta' \) is the average dynamic viscosity of the fluid. This creates velocities of the form,
These instantaneous velocities can be used to determine where a particle would move through the channel. A particle at current position \((z_0, x_0)\) will have a new position of \((z_n, x_n)\) defined by,

\[
\begin{align*}
  z_n &= z_0 + t_{\text{step}} \frac{dz}{dt} \\
  x_n &= x_0 + t_{\text{step}} \frac{dx}{dt}
\end{align*}
\]  

where \(t_{\text{step}}\) is a small time step that can be chosen to be small enough to give accurate measurements of the particles’ movement for the current velocity. Typical values range from \(1 \mu s\) to 1 ms. This process can be repeated in order to determine a trajectory for a particle as it moves down a channel. A sample trajectory for a particle in a loss based trap can be seen in Figure 4-2.
Figure 4-2 Trajectory of bead in loss based trap. Two equal forces causing particle to trap in center of channel

The particles motion would be described by this simple situation if the channels were not present. In reality the channels can interact with the particle’s motion. At an intersection the sum of forces may cause the particle to collide with the walls of the channel. In order to account for this, the particle’s motion needs to take into account what happens if the wall is present. After a particles location is determined the program checks to see if the particle is within the bounds of the channel. If the values falls outside the channel a new value is calculated taking into account the magnitude and direction of the forces so that the particle will move along the channel wall in the appropriate direction. Due to the damping of the liquid and the small effects of the collision the particle will continue along the wall rather than bounce off.

These speeds and positions are accurate in situations in which there is no fluid flow and in fluids with viscosities where the Brownian motion is insignificant over the region of interest. In order to get a more accurate picture of a possible particle
trajectory it is necessary to factor in fluid flow and Brownian motion. Brownian motion is described by,

\[ \overline{d^2} = 2Dt_{\text{step}} \]  

(3.4.5)

where \( \overline{d^2} \) is the mean square distance of the particle movement, and D is the diffusion coefficient. The diffusion coefficient can be found with the Stokes-Einstein equation for low Reynolds numbers of laminar flow,

\[ D = \frac{k_B T}{6\pi \eta' r} \]  

(3.4.6)

where \( k_B \) is Boltzmann’s constant and T is the temperature of the liquid. This gives an average displacement of,

\[ d = \sqrt[3]{\frac{k_B T}{3\pi \eta' r} t_{\text{step}}} \]  

(3.4.7)

This displacement is independent of direction, and therefore the change in particle position due to Brownian motion is defined by,

\[ z_b = d \cos(\theta) = \cos \theta \sqrt[3]{\frac{k_B T}{3\pi \eta' r} t_{\text{step}}} \]  

\[ x_b = d \sin(\theta) = \sin \theta \sqrt[3]{\frac{k_B T}{3\pi \eta' r} t_{\text{step}}} \]  

(3.4.8)
where $\theta$ is the angle of the Brownian motion in relation to the z axis which is randomly generated by the computer. If included in the calculations this creates a new formula for position $(z_n, x_n)$,

$$\begin{align*}
z_n &= z_0 + t \frac{dz}{dt} + z_0 = z_0 + t \frac{dz}{dt} + d \cos(\theta) \\
x_n &= x_0 + t \frac{dx}{dt} + x_0 = x_0 + t \frac{dx}{dt} + d \sin(\theta)
\end{align*}$$

(3.4.9)

Since each situation will yield a different random movement no two trajectories with Brownian motion will be identical. The trajectory of a particle in a loss based dual beam trap is shown with Brownian motion present in Figure 4-3.

![Figure 4-3 Trajectory of a 0.5μm particle with Brownian motion](image)

For a simple straight channel the pressure differences across the channel can be used to determine the flow speeds. The fluid flow in a channel is not, however, constant throughout the channel. The fluid interaction with the wall forces the fluid at the wall to have a velocity of zero; the fluid in the center has the maximum flow velocity. The speed of the liquid in the majority of the positions within the channel is
affected by the fluid that touches it on both sides, see Figure 4-4 for an example of a flow profile. This differential equation can be easily solved for a round channel, but in rectangular channels this is more difficult. As intersections are added, it becomes necessary to determine flow velocities numerically.

![Figure 4-4 Flow profile in a straight liquid core](image)

In order to get accurate flow profiles the program COMSOL is used as a finite element analysis package. Microchannels have laminar flow, described by the Navier-Stokes equation and the continuity equation,

$$
\rho \left( \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \right) - \nabla \cdot \vec{\sigma} = \vec{b}
$$

$$
\nabla \cdot \vec{u} = 0
$$

(3.4.10)
where \( b \) is the volumetric or body force, which can include gravity and/or electric forces that might cause fluid motion, \( u \) is the velocity of the fluid, and \( \rho \) is the density of the fluid. The total stress tensor is defined as,

\[
\sigma = -p\mathbf{I} + \mu' \left[ \nabla \bar{u} + (\nabla \bar{u})^T \right]
\] (3.4.11)

Where \( \mu' \) is the viscosity of the fluid, \( \mathbf{I} \) is the identity tensor, and \( p \) is the pressure.

[83] Boundary conditions can be applied as pressure differences between inlets and outlets as well as initial velocities at inputs and outputs. This allows for an accurate flow profile to be created, Figure 4-5 shows a flow profile of an intersection.

![Figure 4-5 Intersection flow profile at an intersection with 10\( \mu \)m/s upwards and 40\( \mu \)m/s to the left](image)

Figure 4-5 Intersection flow profile at an intersection with 10\( \mu \)m/s upwards and 40\( \mu \)m/s to the left
Flow is additive, so if channel one has an input 1\( \mu \text{m} / s \) and channel two has an input of 1\( \mu \text{m} / s \) there will be an output of 2\( \mu \text{m} / s \). Flow profiles are scalable: a flow profile for a 2\( \mu \text{m} / s \) flow is the same as 2 times a flow profile from of a 1\( \mu \text{m} / s \) flow. This allows for each input to be simulated separately and combined and scaled as needed in Matlab. Flow profiles for each input were created separately in COMSOL and then exported as matrices. These matrices were then imported into Matlab where they are manipulated to fit the specific flow parameters being used.

In order to find the effect of the flow on the particle the Stokes equation (3.4.2), can be used. In steady state without any external forces other than from, the flow the particle will slowly speed up until it moves at the speed of flow, at this point the particle will no longer feel a force. Stokes equation can be written as,

\[
\begin{align*}
F_{\text{stokes}}(z) &= 6 \pi \eta r \left( v_z - \frac{dz}{dt} \right), \\
F_{\text{stokes}}(x) &= 6 \pi \eta r \left( v_x - \frac{dx}{dt} \right)
\end{align*}
\]

(3.4.12)

where \( v_z \) and \( v_x \) are the fluid velocities along \( z \) and \( x \) respectively. This results in a particle velocity of,

\[
\begin{align*}
\frac{dz}{dt} &= \frac{F_z + F_x}{6 \pi \eta r} - v_z, \\
\frac{dx}{dt} &= \frac{F_x + F_x}{6 \pi \eta r} - v_x
\end{align*}
\]

(3.4.13)
By combining both the forces from the fluid flow and those of the optical beams complex motion can be simulated. Figure 4-6 shows a velocity magnitude plot with black arrows to demonstrate the direction of the local motion for a $3\mu m$ particle. This situation has a $10\mu m/s$ flow leaving from the top and a $40\mu m/s$ flow leaving to the right as well as an 30mW laser pointing along x. The trajectory shown in white is the trajectory of a $3\mu m$ particle as it moves through the intersection.

Figure 4-6 Combined flow and optical force simulation showing bead velocity field and simulated trajectory. Flow as in Figure 4-5 with 30mW laser acting upwards.

4.2 Particle Tracking

Videos of particle trapping and manipulation experiments are recorded with an Andor Luca S camera. The videos are used to analyze the results and don’t affect the
outcome since they aren’t part of a feedback loop. For the majority of trapping and manipulation experiments particles range from 0.5\(\mu m\)-3\(\mu m\). Size is limited by the power requirements and the height (5\(\mu m\)) of the channel respectively. The main requirements for a tracking program are to be able to detect a particle from the background, to recognize which particle is which, to create a set of position data for each particle, and to record statistics such as velocity, particle size and in some cases destination.

In my tracking software I used two different techniques for detection, depending on the amount of movement of the particles (i.e. for particles trapped throughout, and for particles moving through the channel). For particles that are moving throughout the video or that have no overlap in their starting and ending locations, the contrast of the particle with the background can be increased by background subtracting. Though any background frame can be chosen, better results are possible when there are as few particles as possible in the channel. Anytime there is a particle in the same location as in the background the particle will be “lost” by the program. In Figure 4-7a, a frame is shown with a small 0.5\(\mu m\) particle (marked in red). Although the particle can be identified by eye it’s difficult to see compared with other stationary things. When compared with the background (see Figure 4-7b) with no beads present it’s possible to see the change but very difficult for a computer to detect. When the background is subtracted from the frame of interest (see Figure 4-7c), the image still contains noise but most of the channel and other objects are removed. The particle becomes easier to identify but still is shrouded in noise. At this
point the program “blacks” out the parts of the frame that are not part of the channel (see Figure 4-7d). This allows for quicker processing, smaller file sizes, and fewer artifacts from slight changes in the channel’s position in the frame. Next, the program digitizes the image to black and white, values above a certain level are made white and those below are made black. For this particular video a threshold value of 0.6 was used, this means that the lightest 40% of the frame is made white (see Figure 4-7e). Depending on the contrast of the video, the amount of light illuminating the

Figure 4-7 a) Frame of interest with 0.5µm particle b) background frame c) Frame of interest with background subtracted d) sections of image that are not the waveguide are removed e) Converted to black and white f) small white pixel sections removed

54
sample, and the size range of the particles this number varies from about 0.5 to 0.8. This gets rid of a large amount of the noise and reduces the size of the spots of noise. In most cases, the noise takes up only a few pixels whereas a particle, even a small one, takes up tens of pixels. Matlab is used to remove white sections that contain fewer than 15 pixels (see Figure 4-7f). This value seems to get rid of the noise the majority of the time while still identifying the 0.5μm particles. Larger values can be used for situations in which only large particles are present, reducing the chances of noise being misidentified as a particle. To make it easier to estimate particle size black spots within white sections are filled. This process creates a new black and white image that contains only particles.

Though particles are easy to see it’s necessary to convert the information from the image into data. This was done with a Matlab function called regionprops (), which gets information about “connected objects” in a black and white image. In this case each particle represents an object. Regionprops is able to determine the particle’s area, the radius along the shortest distance, the radius along the longest distance and the centroid of the shape. The radius and area values can be used to determine what size particle is being tracked as well as to make sure an object is a particle. Since particles should be round, a long skinny object is likely not a particle and therefore the program can ignore that object. These values can also be used to try and determine if the object is multiple particles stuck together by seeing if two circles can be fitted to the data.

55
The main information pulled from the frames by regionprops is the centroid. This tells the software where objects are located in each frame. This information is passed to a function called, track, written by Daniel Blair and Eric Dufresne [84]. This takes the particle locations and compiles tracks for each of the particles. Minimum track length and how long a particle can disappear can be adjusted to reduce false trajectories from noise and connect trajectories if a particle is “lost” during the background subtraction. A set of trajectories from the video with frame image shown above and a flow field of the same situation is shown in Figure 4-8.
Figure 4-8  a) Trajectory and b) flow field profile of intersection with no flow into the upper channel and no laser

For situations in which the particle is trapped throughout the video subtracting the background doesn’t yield good results because the bead’s current position gets subtracted out with the background. In this case it’s necessary to locate the approximate particle position and mask positions without beads so that the noise from these sections doesn’t effect the detection. Large beads such as the 3μm beads have
white centers and, therefore, can be fairly well found simply by adjusting the black and white conversion level such that the centers are white and the outside of the beads and the background are black. Small beads such as 0.5μm beads appear dark on the screen, in this case it’s necessary to invert the image coloring. It’s usually possible to determine an ideal black and white conversion level for the sections of interest. This can then be passed to ‘regionprops’ and tracking is done by the same technique.
5 Orthogonal Trapping

The majority of trapping techniques require some level of focus in the beams, such as the single and dual beam traps mentioned earlier. It’s very difficult and impractical to create a single beam trap on chip due to losses associated with curves and changes in dimensions of channels. Therefore, waveguides typically produce collimated beams. These leaves two simple options, a dual beam trap like the loss-based trap mentioned in the background section where the two beams act opposite of each other, and one that has two orthogonally propagating beams. In this chapter, I introduce this orthogonal beam trap (OBT) for particles with dimensions on the scale of a micron where in the past they have only been used in the Rayleigh regime [85]–[87] which is governed by different criteria. [88]

Ashkin originally introduced two trapping schemes, the dual beam trap (DBT) and the single beam trap (SBT). The SBT has its scattering force balanced by its gradient force (see Figure 5-1a) and the DBT has opposing scattering forces (see Figure 5-1b). The OBT instead has the scattering force from one beam counteract the gradient force from the second and vice versa (see Figure 5-1c). Intuitively it seems like this orientation would lead to the particle moving along the diagonal, in reality there is an increase in the gradient force from beam 1 as the particle is pushed from the center by beam 2, this reduces the net force trapping the particle. The asymmetry of the gradient force gives rise to one location in which the two balance. Though an optical beam is Gaussian at a specific point along its beam waist it is a constant value along its propagation direction (given low loss). This is shown in Figure 5-1 as the
point $x_\gamma$. The forces along the z direction are qualitatively shown in Figure 5-1d, where the vector sum of $\vec{F}_{\gamma 1}$ and $\vec{F}_{\gamma 3}$ create a trapping potential at $z_\gamma$. This same concept applies along the x direction.

By assuming two identical Gaussian beams of equal power this situation can be analyzed more quantitatively in the ray optics regime. The Gaussian beams take the form of,

$$I_1(x) = I_s e^{\gamma x^2}; \quad I_2(z) = I_s e^{\gamma z^2}$$

(3.5.1)

where the beam parameter is $\gamma = 2 / \omega_n^2$ and $\omega_n$ is the Gaussian beam waist. This leads to scattering and gradient forces of the form,
where $k_g$ and $k_s$ are constant coefficients, $A$ is the cross section of the particle, $c$ is the speed of light in a vacuum and $Q_g$ and $Q_s$ are the traditional trapping prefactors used with ray optics [49]. To balance the forces along the $z$ direction it is necessary that,

$$F_{g1}(x) = F_{g2}(z)$$  \hspace{1cm} (3.5.4)$$

and that in the $x$ direction,

$$F_{s1}(x) = F_{s2}(z).$$  \hspace{1cm} (3.5.5)$$

The equilibrium position of the OBT can be determined by solving equation (3.5.4) and (3.5.5) for $x_T$ and $z_T$,

$$x_T = z_T = \frac{k_g}{k_s} \frac{1}{2} \frac{k_s}{k_g} \frac{\omega_0^2}{4}.$$  \hspace{1cm} (3.5.6)$$

This trapping location holds true regardless of power, as long as the two beams have equal power. Raising the scattering force will also raise the gradient force equally, allowing the trapping point to remain the same. This condition holds true in two locations, only the location where the gradient force is restoring will trap. This can
be explained as the trapping point must be closer to \( z = 0 \) than the maximum gradient force,

\[
x_{z,\text{tr}} = z_{z,\text{tr}} = \frac{1}{\sqrt{2\gamma}} = \frac{\omega_0}{2},
\]

making trapping only stable when,

\[
\omega_0 \leq \frac{\omega_0}{2} \frac{k_c}{k_c}
\]

This means that trapping is possible even with collimated beams. Though it is necessary for the beam to be narrow enough to have a strong gradient force that the gradient force is able to counteract the scattering force from its opposing beam. The most confusing part of this trap is the dependence of the scattering force on the coordinate that is transverse to the direction of the beam. For example, Beam 1 propagates along the \( z \) direction, but the scattering force of interest is dependent on the \( x \) position \( F_{\text{sc}}(x) \), because of this the scattering force felt by the particle is less than its maximum possible value. In Figure 5-2 the scattering force along \( x \) for Beam 1 is plotted with the gradient force along \( z \) for Beam 2, showing that the scattering force must be less than its maximum to be balanced by the gradient force.
Figure 5-2 Analytically calculated forces. (a) Scattering force of Beam 1 along x and gradient force of Beam 2 along z. The arrows mark the range in which a trapping point must occur and are used to create the gradient force and scattering force maximums in (b).

In order to get a better picture of this trapping scenario a specific example of a collimated Gaussian beam will be used ($\omega_n = 0.75 \mu m$, $\lambda = 532 nm$, and $P = 60 mW$) as well as specific parameters for a microparticle (index $n = 1.4$ and diameter $d = 1 \mu m$). These parameters were used to calculate Figure 5-2 and throughout the majority of the rest of this section. The majority of the plots and analytical solutions were calculated using the simulation technique mentioned previously in chapter 4.1. Equations (2.2.4) and (2.2.5) for loosely focused and collimated beams [50] were used and numerically integrated. No flow was used in the channel so the motion of the particle was only dependent on the forces from the two beams. The two arrows in Figure 5-2a mark the locations of the maximum gradient and the maximum scattering forces, these points are then plotted versus particle size in Figure 5-2b. The symbols represent the forces calculated using the beam parameters and simulation program, the lines are parabolic fits based on equations (3.5.2) and (3.5.3) showing a good fit (R value>0.98) for $d < 2 \omega_n$. The stability condition from equation (3.5.8) holds.
when the scattering force at the maximum gradient force (red line in Figure 5-2b) is below the maximum gradient force (blue line in Figure 5-2b), which hold true at $d = 0.44 \mu \text{m}$ for our parameters. Though the simple parabolic fit is less ideal once the particle is of similar size as the beam waist due to the fact that the beam is smaller than the particle and thus the power is constant, the conditions for stability still hold.

Dynamic simulations were performed and used to calculate the specific trajectory, forces and other values for our parameters. Diffusion was ignored since changes in the particles position were much smaller than the diameter of the particle used. The trajectory (see Figure 5-3a for a scaled plot) shows the particle being pushed along the center of Beam (red) 1 by its scattering force towards the intersection where it gets pushed slightly from the scattering force of Beam 2 (blue) until it is trapped by the gradient force of Beam 1. From this simulation the trapping point is determined to be slightly off center at $(x, z) = (0.1 \mu \text{m}, 0.1 \mu \text{m})$. The trapping point can also be seen in Figure 5-3b, which shows the total net force, $|F_x| + |F_z|$, at any given location within the intersection. Therefore, the trapping point should be the
only position with both $F_x$ and $F_z$ equal to zero. Side and top are views shown to make the trapping point easier to see. The four peaks that surround the trapping point are the local maxima of the gradient force for the two beams.

The particle velocities in the $x$ and $z$ directions are shown in Figure 5-3c. The particle is initially pushed along Beam 1 by the scattering force, which creates a constant $z$ velocity. As it enters the intersection it comes into Beam 2 and initially experiences an increase in its $z$ velocity due to the gradient force from this second Beam. After passing $z = 0$, the particle will start to slow down due to the gradient force of Beam 2 now acting against the scattering force Beam 1. This continues until the two forces acting on the $z$ direction balance. Due to the loading of the trap initially coming along Beam 1 the $x$ velocities are slightly different. In this case the velocity is initially zero since the particle is already in the center of Beam 1 and therefore experiences no effect from the gradient force. As it enters Beam 2 it starts to feel the effect of the scattering force from this beam, causing it to initially speed up, pushing it out of the center of Beam 1 and thus causing a counteracting force from Beam 1 slowing the particle back down until the two forces reach an equilibrium.
In order for there to be stable trapping the trapping potential must be deep enough. The optical potential at \( x = x_\gamma \) along \( z \) (see Figure 5-4a) was calculated by integrating the sum of \( F_{z,2}(z) \) and \( F_{z,1}(z) \) along \( z \) (see Figure 5-2a). The overall slope of the potential is caused by the constant value of the scattering force \( (F_{z,1}) \) along \( z \) whereas the well is caused by the gradient force from beam 2. The potential well has a confining depth of \( \Delta W = 1.9 \times 10^{-18} \) J = 12eV, which is sufficient to create a stable trap.

The stability of the trap was found for a range of particle diameters and beam waists with all other parameters kept the same (see Figure 5-4b). This plot is the equivalent of equation (3.5.8). The linear relationship between the two parameters shows that in order for stable trapping to occur the beam waist must be no more than 2.4 times the particle diameter with our parameters. For beams with a beam waist larger than this value the gradient force is insufficient to overcome the scattering force and the particle will deviate from its path, depending on the relative strengths of the scattering forces at its new position the bead will then take the path of one of the scattering forces (see Figure 5-5). These results are only approximate for particle diameters similar to \( \lambda \) since these value are within the Mie regime.

The OBT can be used both in free space as well as on chip. In order to achieve the proper beam waist, for a weakly focused Gaussian beam of wavelength \( \lambda = 532 \text{nm} \), the beam would need a divergence angle of 9.7° and a focal depth of 8.84 \( \mu \text{m} \). This would result in an increase of only 0.026% from the minimum beam
waist to the trapping location at \((0.1\mu m, 0.1\mu m)\) and therefore can be considered collimated over the region of interest. This trapping technique is ideal for on chip trapping since many conventional traps can’t be used due to their requirements of divergent beams. The fundamental mode of a planar waveguide is an excellent approximation for a collimated beam and liquid-core waveguides can be produced with mode diameters of 0.5-3\(\mu m\) that are large enough for microbeads.

![Figure 5-5 Trajectory of 3\(\mu m\) bead with 5\(\mu m\) beam waist which is unable to trap due to having a diameter greater than 2.4 times the beam waist.](image)

In conclusion, I’ve introduced a new trapping technique for microparticles that is performed with collimated Gaussian beams. It requires two orthogonal beams; the scattering force from one beam counteracts the gradient from the other and vice
versa. This trap is ideal in that it can trap with collimated beams, it’s self-loading, and it’s easily able to trap particles with low indices of refraction. The majority of biological particles have low indices of refraction and traditional trapping techniques create a large amount of heat do to their highly focused beams. The stability condition which for a given beam size only allows particles of a certain size or lower to be trapped offers the capability to selectively trap only specific particles. This trap features unique characteristics allowing for trapping both on chip and off.
6 Particle Sorting

In this section, I will introduce an approach to sorting which uses a combination of optical and fluidic forces to selectively remove particles from a channel based on size. We designed this sorting scheme to work with two different arrangements. One of these arrangements allows for easily tunable control over a size range whereas the second offers the ability to remove 100% of a certain size from a mixture of beads. Unlike many traditional sorting techniques like flow cytometry this technique is automatic and is defined by the optical beam parameters and the speed of the flow, no user input is required. [89]

![Figure 6-1 (a) Photo of actual and (b) simplified schematic of “H” mask chip](image)

Traditionally we’ve used ARROWs with a single liquid core that creates an S-shape since it was necessary to remove particles from the main fluidic channel we designed a new chip. This chip has a shape similar to that of the letter “H” (see Figure 6-1). It features four reservoirs epoxied on to each end of the liquid-core waveguide in order to allow flow control and particle introduction on all ends. By using only three of the four reservoirs of this chip we can create a T-intersection so that certain
particles can be separated from bulk. The liquid-core ARROW channels are 5x12μm² in cross section and have alternating silicon oxide and silicon nitride of thicknesses 3196/12/300/142/285/127nm (SiO₂/SiN/SiO₂/SiN/SiO₂/SiN) above the liquid core and layers of 270/93/270/93/270/93/285/127nm (SiN/SiO₂/SiN/SiO₂/SiN/SiO₂) under the core. Green laser light (532nm) form a (Lighthouse Photonics Sprout) was coupled into the solid core by butt coupling a single mode fiber to the edge of the solid core. A mixture of Invitrogen sulfate latex beads (r=0.25μm, 0.5μm, 1μm and 1.5μm) were used in tests. The particles were suspended in a water-based solution containing 0.05% tween 20, 1mM sodium azide, and a 8% w/v mixture of beads contain equal concentrations of each size. Flows were created between the reservoir containing the bead mixture and output reservoirs via hydrostatic flow or by a Harvard Apparatus syringe pump.

![Figure 6-2 Instantaneous velocity vectors from optical forces of a (a) 0.25μm and (b) 1.5μm bead](image)

The idea behind this sorting technique is the different size dependencies of the forces acting on a particle in the channel. The particles are controlled fluidically through pressure-based flow and obey Stokes’ law as given by equation (2.3.2) can be written in terms of the flow speed \( v \),
\[ F_F = 6\pi\eta vr. \]  

(3.5.9)

The scattering force of equation (2.2.1) can be rewritten in terms of intensity, \( I \), as

\[ F_L = Q\pi \frac{r^2}{c} I. \]  

(3.5.10)

The differences in optical force can be seen in Figure 6-2 where the optical forces from a 25mW beam act on particles with radii of 0.25\( \mu \)m and 1.5\( \mu \)m without the presence of flow. It can be clearly seen, in Figure 6-2, that the gradient force is stronger on the larger particle, as the arrows point more towards \( x=0 \).

![Figure 6-3 Several flow trajectories simulated with Brownian motion](image)

The dependence of the optical force on the particles size, \( F_L \), is quadratic in nature whereas on the force of the fluid flow, \( F_F \), it is linear, this means that we can scale the relative values of the two forces and adjust the direction of the larger beads.
compared to that of their smaller counterparts. The large beads feel a larger impact from the laser and thus follow its path, and the smaller particles which don’t feel enough effect from the laser will follow the direction of flow.

In order to obtain accurate analysis dynamic simulations were used to find the particle trajectories. The simulations were performed using the simulation technique mentioned earlier in chapter 4.1. Since the center of the channel experiences different optical and fluidic forces, the positions of the particles in the channel determines both the trajectory of the particle and its final destination. A single straight channel has a fairly easily defined velocity curve but the intersections of our “H” chip added complexity to these velocities. We used COMSOL to determine the velocities, \( \mathbf{v}(x,y,z) \), within the channel depending on relative flow speeds of the four input channels. These flow speeds were then used to determine the forces due to the fluid flow, \( \mathbf{f}_f \). The mode profiles of the liquid-core ARROWS were used in conjunction with position dependent optical forces formulas for loosely focused and collimated beams [50] to determine the forces due to the laser, \( \mathbf{f}_l \). The instantaneous velocity of the particle can be determined by finding the vector sum of these forces. Using the particle’s current position, its instantaneous velocity, and a time increment of 1ms, the next position was found. Although there was some difference in actual trajectories due to Brownian motion (see Figure 6-3) we found very little difference in the actual percentage of particles that sorted, where sorted is defined as particles which end up in a different channel than they would without the
optical forces present, and therefore ignored Brownian motion for the trajectories. Particles were tracked via overhead videos and compared to simulated trajectories. The sizes of the particles were found via the cross-sectional area and used to find percent sorting values.

Since this sorting technique relies on the relative flow and laser power combinations, various combinations were tested and then analyzed to determine how well each size of bead was selected. We looked at two cases; in both the particles initially travel with the fluid flow, and the scattering component of the optical force removes the particles from the flow. The gradient force and the scattering force affect the exact trajectory of the particles. Since both components of the optical force are dependent on position of the particle in the channel and bead size this leads to a strong variation in actual trajectory of the beads.

6.1 Orthogonal Sorting

The first layout we considered has the flow orthogonal to the laser's propagation direction (see Figure 6-4). The particles are initially located in reservoir 1. Without the effects of the laser the flows are set up such that most of the flow goes from reservoir 1 to reservoir
2, and the flow to reservoirs 3 and 4 are minimized (see Figure 6-5). The laser therefore points down the leg of the T towards reservoirs 3 and 4 and particles which travel in this direction are “sorted”. The larger particles will be pushed towards reservoirs 3 and 4, while the smaller particles will continue with the flow into reservoir 2. The combined effects of the flow and laser can be seen in Figure 6-6a where the velocity vectors of 0.25μm (gold, not sorted) continue straight whereas the 1.5μm (blue, sorted) turn and take different paths. Specific tracks of 0.25μm, 0.5μm and 1μm beads can be seen in Figure 6-6b where the experimental results are shown as points whereas the simulated trajectories are shown as lines. The agreement between the two is good and differences are most likely due to the effects of Brownian motion.

In this case a bead is considered sorted if \( x(z = -6\mu m) > 6\mu m \). Due to the particle’s position in the channel, the variations of flow speed, and optical mode
intensity in the channel, the lateral position of the bead in the channel as it approaches the intersection directly effects whether the particle is sorted or not. A particle that approaches the intersection along the \( x = 6 \, \mu \text{m} \) side will quickly be sorted even if it is a small particle. On the other hand, a particle moving along the opposite side of the channel, \( x = -6 \, \mu \text{m} \), has to move a large distance along the x direction in order to be sorted. This effect is present for all bead sizes but particularly strong with the small, \( 0.25 \, \mu \text{m} \), beads as seen in Figure 6-6c.

Figure 6-6 Orthogonal Orientation (0.25\( \mu \text{m} \) is gold, 0.5\( \mu \text{m} \) is red, 1\( \mu \text{m} \) is green and 1.5\( \mu \text{m} \) is blue) (a) Velocity vector field with flow and laser. (b) Flow trajectories comparing experimental (dots) and calculated (lines). (c) Simulated trajectories with \( v_x=10 \, \mu \text{m/s} \), \( v_y=0 \, \mu \text{m/s} \) and \( P=20.1 \text{mW} \). (d) Sorting efficiencies for \( v_z/v_x=0.3 \).

We defined the sorting efficiency as the ratio of the number of beads of a particular size that are sorted to the total number of these beads. By varying the
relative flow speed and laser power we found the percent sorting efficiency for all bead sizes over a range of power to laser flow velocity ratios (see Figure 6-6d). The level of flow into the cross channel ($v_\gamma > 0$) increases the number of beads being sorted and therefore increases the efficiency. The theory lines are not smooth due to the limited number of particles and lateral positions we used in the simulation. Due to this finite level of flow along the $X$ direction there will always be particles sorted even with no laser power present. This minimum value of particles sorted is approximately the same for all bead sizes. The concentration of a specific size of particles above this initial value can be easily tuned making the technique ideal if a specific concentrations is wanted, as long a complete removal of species isn’t required.

### 6.2 Counterpropagating Sorting

In this second orientation, the chip is rotated so that the laser is able to propagate along the liquid core channel between reservoirs 2 and 1 (see Figure 6-7). In this case the flow is designed to move beads from reservoirs 3 and/or 4 towards reservoir 2. This creates a small flow from reservoir 1 to reservoir 2 (see Figure 6-8). In this case the particles are pushed by
the laser towards reservoir 1 and pulled towards reservoir 2 by the flow. The larger beads will move with the laser towards reservoir 1 and the smaller ones will move with the flow to reservoir 2. It’s also possible to trap particles with this layout by balancing the flow with the laser.

In this case the relative flow velocities and laser power also determine which size bead goes in which direction. In Figure 6-9a it can be seen that regardless of the location in the channel all the 0.25 μm bead velocities lead to the left channel and the 1.5 μm all lead to the right channel. Excellent agreement was found between the simulations and experiments (see Figure 6-9b) where lines are simulation and points...
are obtained from the videos of actual trajectories. In this case a particle is defined as sorted if it reaches a position with \( z > \delta \mu m \). With this layout there is far less effect on initial position to sorting, though the same basic principles apply. This is due to the different flow geometry, and the fact that particles feel a continued optical force throughout their whole flow through the intersection and after. This leads to rounded trajectories with particle centering whether they are sorted or not due to the gradient force from the laser (see Figure 6-9c).

In this geometry it’s possible to completely separate particles of a larger radius from those of smaller (see Figure 6-9d) allowing for complete removal of a certain size from a stream. If a single size or a small range of sizes were desired two intersections of this geometry could be combined to create a “notch filter”

![Figure 6-9 Counterpropagating orientation (0.25μm is gold, 0.5μm is red, 1μm is green and 1.5μm is blue)](image)

(a) Velocity vector field with flow and laser. (b) Flow trajectories comparing experimental (dots) and...
calculated (lines). (c) Simulated trajectories with $v_z = v_x = 10 \mu m/s$ and $P=25 mW$. (d) Sorting efficiencies for $v_z/v_x=0.7$.

In summary, I have introduced a technique and two designs for automatic, no feedback needed, sorting on an ARROW chip. This technique uses a combination of pressure-based flow and optical forces to select certain sized particles. The relative size and cutoff sizes can be tuned by adjusting relative levels of fluid flow speed and optical power. This technique can also be used to sort by other parameters such as refractive index (see Figure 6-10), since this is another parameter that has a different dependence optically than fluidically.

Figure 6-10 Simulated particle trajectories for a range of indices for a 3μm particle with $v_z=20 m/s$ and $P=30 mW$
7 MMI trapping

In this section I will introduce a novel trapping technique which uses a MMI to produce multiple trapping spots, with one input. The number of spots and their locations of the spots is spectrally tunable. This trap can be used to trap multiple particles simultaneously or as a “conveyor belt”. [90] In order to create this trap a new ARROW chip was designed with an MMI section (see Figure 7-1a). Particles can be introduced to the liquid-core ARROW through one of two reservoirs and can be moved via pressure based flow from one side of the chip to the next. We used a Lighthouse Photonics Sprout laser with a wavelength of 532nm to pump a Del Mar Photonics Trestles laser, which can be continuously tuned from about 741nm to 879nm. This allows for a single mode optical fiber to be coupled, but multiple wavelengths to be tuned to. The MMI is designed to be 75μm wide and 1.9mm long.

![Figure 7-1](image.png)

**Figure 7-1** (a) Schematic of MMI chip and (b) SEM of chip

The actual sizes vary by up to 1.6μm, depending on fabrication (see Figure 7-1b). Though multiple inputs are present on one side of the chip, by using the central waveguide, a single central input can be used from either side of the chip. The light coupled into the solid core enters the MMI at the central waveguide and produces a
spot pattern that depends on the input wavelength as described in section 2.5. If the wavelength is tuned to an ideal value, the higher the wavelength the lower the number of spots. These waveguides were designed to produce 7 spots with a HeNe laser, so

![Simulated MMI spot pattern for a range of wavelengths](image)

Figure 7-2 Simulated MMI spot pattern for a range of wavelengths

the ideal spot wavelengths that fall closest to the range of our laser are 738.2nm (6 spots) and 885.9nm (5 spots) but since these are slightly out of the range of our laser, I adjusted the wavelength of the laser until we had sufficient power but still had the correct spot pattern. The values used were 753nm and 876nm. The power at these wavelengths is about 120mW before coupling. I found that with the length and wavelength range I’m using that within about 20nm either direction of the ideal wavelength the correct number of spots was present but the intensity distribution was less evenly distributed (see Figure 7-2) causing the trapping potential at each spot to be more varied.
In order to verify the MMI pattern within the intersection of the MMI and liquid-core waveguide we filled the channel with DyLight830 a near-IR dye with a broad range of absorption/fluorescence (see Figure 7-3a) and an Edmund Optics Inc. 900nm long pass filter to block the input wavelength. When the dye is illuminated an Andor Luca top view camera can be used to image the fluorescence (see Figure 7-3b and Figure 7-3c). The spot pattern compared to the pattern predicted with simulations can be seen in Figure 7-4. There is an overall Gaussian like background that is particularly noticeable on the 876nm spectrum (see Figure 7-5). This is from back reflections throughout the chip, when light hits a layer such as the transition between solid and liquid core some of the light is reflected back. Figure 7-5 shows the measured spectrum compared with a simulation in which all material changes reflect and one in which they absorb. Depending on the wavelength and materials this reflection can cause an overall background. By fitting Gaussians to each peak of the spot patterns it was found that for 753nm the full width half max (FWHM) varied from 4.1μm to 5.3μm and the 876nm FWHM ranged from 4.7μm to 6.2μm when the initial input solid core mode was measured to be about 3.9μm. Overall, the more
spots the narrower each spot. The average spacing of the spots for 753nm is 13.03\,\mu m with a standard deviation of 0.77\,\mu m, and for 876nm the average is 15.51\,\mu m with a standard deviation of 0.69\,\mu m.

![Figure 7-4 Simulated (lines) and measured (dots) spectra for 876nm and 753nm](image)

In this case the actual trapping is very simple. A single laser is coupled into the waveguide and propagates through the MMI turning into multiple spots at the intersection. This produces the equivalent of 6/5 lasers all shining parallel to each other. Each spot produces a scattering force that pushes the bead into the wall and a gradient force that keeps the particle in the beam and from moving with the flow present in the channel. In this case the gradient force counteracts the force from the flow in y direction and the scattering force is balanced by the force from the wall in the x direction. In theory a dual beam trap should be possible but due to inconsistencies in manufacturing the relative power of each spot varies. Thus, if one spot is perfectly balanced the rest of the spots aren’t causing particles to be pushed into one or the other of the walls depending on relative intensity.
Each spot can hold one or more particles, depending on the size of the particles and the power of the spot (see Figure 7-6). If the power isn’t sufficient to trap a particle the flow will cause the particle to continue to the next spot. If an exact quantity of particles at each spot is not required, the power can be left on and at a single wavelength and the trapping locations will slowly fill. Once the first spot fills, 2-3 particles depending on power, one of its particles will be pushed out and get trapped in the next position. Once the second position fills the beads will move on to the third, and so on. This allows for constant analysis and acts as a form of concentrator that only allows a small number of particles.

Figure 7-6 Multiple particles trapped

Another method of filling the trap is by using a shutter to turn the laser power on and off. A single bead comes down the channel and is trapped by the first spot. At
this point the first spot can either be filled to capacity or with only one bead in this spot. Before the next bead comes the power can be turned temporarily off until the beads from the first spot make it out of the area of influence of the first spot, then turn the laser back on. The flow will continue to move the particles into the second spot’s influence, where they will be trapped. This technique can be repeated until all spots are filled and can be continuously used to fill the spots with new beads. One challenge with this technique is that depending on the beads’ position in the channel it will have a different level of interaction with the walls and therefore a different flow speed, which can make it difficult to both clear the old spot and not pass the new spot on all trapping locations at once.

![Image](image.png)

**Figure 7-7 Interwoven spot patterns of 876nm (orange) and 753nm (red)**

The third technique of filling the trap is by using the locations of the trapping points of multiple wavelengths to allow the particles to be slowly moved down the channel. The first spot location a bead would interact with is that of 753nm (6 spots) where the bead will be trapped (see Figure 7-7). When one wants to move that bead instead of turning off the power the power can be shifted to 876nm (5 spots). Since the particle is no longer trapped in its current location it will be picked up by the flow and moved into the first spot of the 876nm pattern. If the laser is then moved back to
753nm (6 spots) the particle will again no longer be trapped and move to the next trapping location. This pattern continues for all spots along the track. This can be used to fill all the positions or to move one or several particles along the trapping region along a “conveyor belt” (see Figure 7-8).

<table>
<thead>
<tr>
<th></th>
<th>Spot 1</th>
<th>Spot 2</th>
<th>Spot 3</th>
<th>Spot 4</th>
<th>Spot 5</th>
<th>Spot 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_x (nN/m)$</td>
<td>1278</td>
<td>2742</td>
<td>2735</td>
<td>8682</td>
<td>11438</td>
<td>6378</td>
</tr>
<tr>
<td>$k_y (nN/m)$</td>
<td>68</td>
<td>178</td>
<td>480</td>
<td>73</td>
<td>660</td>
<td>1590</td>
</tr>
</tbody>
</table>

Table 7-1 Stiffness of Traps for 876nm

Due to inconsistences in the modes and powers of each spot, every location has a different trap stiffness and potential. Each trapping position has its own values, values for one particular trapping situation can be seen in Table 7-1 and Figure 7-9d. The stiffness of the trap in the x direction ($k_x$) is significantly larger due to the fact that the bead’s position is limited by the wall. This also accounts for the large asymmetry in the trajectory of the bead in the trap (see Figure 7-9c).

Figure 7-8 Conveyor belt screen shots (876nm is orange and 753nm is red)
In conclusion I’ve introduced a photonic technique to create a configurable multi-spot trap on an optofluidic trip. Tuning the wavelength creates well-defined trapping locations which can be used to trap multiple particles or as a “conveyor belt” to move particles along a channel in a controlled manner.

Figure 7-9 Trapped Particles with (a) 876nm and (b) 753nm. (c) Trajectory of first bead trapped with 876nm and (d) potential of same bead.
8 Trapping Assisted SERS Detection

Surfaced enhanced Raman scattering (SERS) offers a large enhancement in small signals making it ideal for situations with low levels of analyte and early detection of viruses and diseases. The majority of SERS techniques currently performed use SERS active surfaces or colloidal metals with large bulk liquids. A few on-chip methods have been performed, with both colloidal particles and with surface modification [91]–[93]. Here I introduce an on-chip technique that uses SERS active microspheres for location specific, controllable SERS detection.

Our technique involves the control of microparticles coated with metallic nanoparticles. Orange fluorescent carboxolate-modified polystyrene beads (FluoSpheres, Life Technologies, 2μm diameter) were used as a framework to attach the nanoparticles. A solution of ethanol, AgNO₃, and butylamine was used to produce a coating of silver on the particle. 20μL of the polystyrene beads were added to 5mL of ethanol, this was then added to a mixture of AgNO₃ (1mM diluted in ethanol) and butylamine (concentration varied, see Figure 8-3). This mixture was then incubated for 50 minutes at a constant temperature which varied from 40°C-70°C. After incubation the particles were vortexed and removed from the silver mixture into ethanol. The silver particles form on the surface of the bead, creating nucleation sites that allow the silver to grow into clumps that coat the particle [94]. One such particle can be seen in Figure 8-1. Note that the microshperes are fluorescently labeled, but this property was not used in our experiments as the fluorescence lies outside the range of the SERS signal and therefore doesn’t affect the results.
In order to test the viability of the SERS particles, Rhodamine 6G (R6G) is used as an analyte. A 12.5µM solution of R6G was produced. This solution was then mixed with a set of beads with Ag coating. As a control sample a second solution was mixed with a set of beads that were uncoated. The mixture was then deposited on a glass microscope slide. An overhead camera was used to find and focus on a single bead. A HeNe laser ($\lambda_{\text{exc}} = 632.8\text{nm}$) was then focused onto the single bead and the back reflection was collected with a 100x 0.85 NA objective lens and directed into a Raman spectrometer (Jobin Yvon). The particles were found to be SERS active; R6G’s SERS spectrum can be seen with the coated particles but not with the uncoated ones (see Figure 8-2).
Figure 8-2 SERS signal from 12.5M R6G solution with coated and uncoated beads

The process of coating the particles can be tuned to produce particles with different sized aggregates and different amounts of coating. Relative concentrations, temperature and time were varied in order to maximize SERS activity and silver coverage. The best conditions found were 60-65°C for 20min with a concentration of 1:0.9 of AgNO₃ and butylamine. A UV/vis was used to measure the absorption of the Ag coated particles (see Figure 8-3). A peak at about 425nm was found which represents the resonance excitation from the Ag nanoparticles.
Figure 8-3 UV/Vis extinction of Ag coated microparticles

The process of coating the particles can create particle groups that have been fused together. As aggregates form on the surface of a polystyrene bead they begin to grow, if they run into another bead they can fuse there, creating a stable link between beads (see Figure 8-4). As the number of beads increase so does the amount of SERS produced. Though each bead produces some level of SERS there is an overall increase that is greater than the sum of the beads making it likely that the combination of beads produces crevices that help increase the overall intensity (see Figure 8-4). Furthermore, the number of particles within a trap can be purposely increased [41], resulting in large concentrations of particles and thus higher sensitivity.
Figure 8-4 a) Particles in groups of 1, 2, and 3. b) R6G intensity is shown relative to the number of beads in a grouping. Inset is taken from height of the peak marked with an arrow.

Additional particles were created by collaborators Mostafa El-Sayed and Mahmoud A. Mahmoud at Georgia Tech. with an attempt to create an increased signal due to the resonance of the excitation on the nanoparticles. These 1μm microparticles were created by a process of swelling [95]. Instead of using aggregate instead these particles were covered with NanoFrames (see Figure 8-5). These gold NanoFrames were produced such that they would have increased resonance at $\lambda_{\text{exc}}$ thus increase the overall signal. Though slightly more SERS was detected the low
levels of NanoFrames on the microparticles meant that there was an insignificant difference from the particles produced with aggregates (see Figure 8-5).

Particles can be found and detected in a bulk solution but are ideal for on-chip detection. Particles were initially tested on chip by introducing a 50µM R6G and 2µM Polyethylene Glycol (PEG) to one reservoir and allowing the particles to flow between reservoirs at about 1cm/s. The PEG was added to the solution in order to
reduce the number of microparticles that stick to the walls of the channel. A HeNe laser ($\lambda_{\text{exc}} = 632.8\text{nm}$) was then focused through a 100x 0.85 NA objective lens onto the edge facet of a solid-core waveguide perpendicular to the liquid-core waveguide in which the particles flow. The majority of the light continues through the channel and a small portion is reflected back up through the objective into the Raman spectrometer (Jobin Yvon) where the excitation light is filtered out. When a particle flows through the intersection, SERS can be detected from the back reflections, see Figure 8-6 for a spectrum of a aggregate SERS particle.

![Figure 8-6](image.png)

**Figure 8-6** (a) Setup used to get SERS spectrum on ARROW (b) SERS spectrum from ARROW waveguide

A loss-based dual beam trap was then employed to trap the SERS coated microparticles. A near-infrared (NIR) beam at 800nm ($\sim 30\text{mW}$) was used to trap the particles along the liquid core. Particles could then be moved in and out of the SERS region. NIR was used so that any SERS produced by the trapping beams would not interfere with those in the visible spectrum, a 532nm laser of low power ($\sim 30\text{\mu W}$)
was then coupled through the solid core to the intersection. Particles within the excitation region should produce SERS, which can be collected via the solid core (see Figure 8-7). Unfortunately, even with large clusters of beads no SERS signal was created. This is most likely due to the Ag heating from the larger powers required to trap, this causes the Ag to melt and produce a smooth coating which is ineffectual at producing SERS. Evidence of this was seen on the overhead microscope display of the spectrometer.

Figure 8-7 Schematic and overhead image of particle in trap

In the future, an ABEL trap can be tried, the ABEL trap can be performed with low optical powers to prevent the Ag from melting. Though the ABEL prevents particle concentrating, the fact that the particles come in fused clumps of varying size means that it’s possible to trap groups of particles to increase the SERS detection.
9 Conclusion

In conclusion, I’ve introduce optical trapping techniques that are particularly well suited to be used on-chip. These techniques will improve the capabilities of small, portable lab-on-a-chip platforms. Specifically, I’ve improved the throughput by reducing the losses at the liquid core – solid core intersections, reduced the fluorescence and background on the ARROW chip in order to improve detection, developed two new trapping schemes and a sorting technique. I also created simulation and tracking programs making it easier to predict and analyze optical and fluidic motion.

I was able to improve transmission through the intersection of the liquid and solid cores by a factor of 17.1 by increasing the thickness of the top oxide from about 4.5μm to 6.5μm. This eliminated the mode mismatch from the crevice created during fabrication. By switching from SiO₂ films to Ta₂O₅ films a large decrease in the photoluminescence was achieved. This reduced the background noise by a factor of 10 and reduced the signal to noise ratio by 12. This made it possible to detect more particles and increases the overall sensitivity of the device.

The simulation software is able to combine optical and fluidic motion and fields in order to produce accurate trajectories and fields within complex intersection and interactions. These matched the results from experiments and can be used in the future to predict particle motion in new situations. The tracking software is able to locate, determine the size, and track a particle across multiple videos within the
presence of both large and small movement for beads ranging from 0.5μm to 3μm. This allows for analysis of trajectories and comparison with simulations.

The first of the two trapping scenarios introduced offers trapping between two orthogonal beams. It can be used with two collimated beams or beams guided in a waveguide, is self-loading and works well with low indices of refraction differences. The self-loading capabilities and the lack of need for focusing make it ideal for on chip applications. The majority of biological particles have indices close to water and this technique is ideal for these situations since this trap works particularly well with small differences in the refractive index of the particle from the fluid.

The second trapping technique uses an MMI to create multiple trapping locations. The number and position of the trapping locations is determined by the laser wavelength making it possible to spectrally configure the trap. These spots can be used to simultaneously trap multiple particles or as a “conveyor belt” to move particles down the channel.

A particle sorting technique was developed which features two designs. Both designs use a combination of optional and fluidic forces in order to select particles based on size, the relative magnitudes of the forces can be tuned to adjust size selectivity. The techniques are automatic and can be tuned to remove all particles above a specific size or to adjust the concentrations of specific sizes.

Lastly, a SERS detection technique using SERS active microparticles is shown. SERS microparticles are created and detected from the channel offering location
specific detection. By trapping these particles they can be introduced to specific locations and can enhance signal as needed.
10 Appendices

10.1 Simulation Software

Simulate_trajectory.m

```matlab
% The tic and toc commands measure the amount of time the simulation takes. % This is useful for simulations that run for a long time, so you can get a sense of when to check on them.
tic

% To run a simulation
simulation0 = true;        % with default parameters
simulation1 = false;        % with beam power ratio
simulation2 = false;        % with range of equal powers
simulation3 = false;        % with range of power ratios
simulation4 = false;        % with range of particle sizes
simulation5 = false;        % with range of x beam waist
simulation6 = false;        % with range of initial x positions
simulation7 = false;        % with range of particle indices

% Simulation 0. Graph a particle trajectory using default parameters.
if (simulation0)
    fprintf ('Simulation 0. Graph a particle trajectory using default parameters.
');
    run default_parameters
    x = initial_x;
    z = initial_z;
    plot_power_hitting_particle_trajectory = false;
    plot_particle_location = false;
    plot_particle_trajectory = true;
    plot_particle_velocity = true;
    draw_waveguide=true;
    run calculate_particle_trajectory
end

% Simulation 1. Graph a particle trajectory for a given beam power ratio.
if (simulation1)
    fprintf ('Simulation 1. Graph a particle trajectory for a particular beam power ratio.
');
    run default parameters;
end
```

x = initial_x;
z = initial_z;
x_beam_power=0.5*z_beam_power;
pplot_power_hitting_particle_trajectory = false;
plot_particle_location = false;
plot_particle_trajectory = true;
plot_particle_velocity = true;
draw_waveguide=true;
run calculate_particle_trajectory
end

% Simulation 2. Graph a particle trajectory for a range of equal
powers.
if (simulation2)
    fprintf ('Simulation 2. Graph a particle trajectory for a range
    of equal powers.\n');
    for i=linspace (10,100,10)*1e-3
        run default_parameters;
        x = initial_x;
        z = initial_z;
        x_beam_power = i;
        z Beam_power = i;
        record_plots             = false;
        plot_particle_position   = false;
        plot_particle_forces     = false;
        plot_particle_x_forces   = false;
        plot_particle_z_forces   = false;
        plot_particle_location = true;
        plot_particle_trajectory = true;
        plot_particle_velocity = true;
        draw_waveguide=true;
        run calculate_particle_trajectory
    end
end

% Simulation 3. Graph a particle trajectory for a range of powers
ratios.
if (simulation3)
    fprintf ('Simulation 3. Graph a particle trajectory for a range
    of powers ratios.\n');
    for i=linspace (0,1,10)
        run default_parameters;
        x = initial_x;
        z = initial_z;
        x_beam_power = i*z_beam_power;
        record_plots             = false;
        plot_particle_position   = false;
        plot_particle_forces     = false;
        plot_particle_x_forces   = false;
        plot_particle_z_forces   = false;
        plot_particle_location = true;
        plot_particle_trajectory = true;
        plot_particle_velocity = true;
        draw_waveguide=true;
        run calculate_particle_trajectory
    end
end

% Simulation 4. Graph a particle trajectory for a range of particle
sizes.
if (simulation4)
fprintf ('Simulation 3. Graph a particle trajectory for a range of particle sizes.
');
for i=linspace (0.25,4,14)*1e-6
    run default_parameters;
    x = initial_x;
    z = initial_z;
    particle_diameter=i;
    record_plots = false;
    plot_particle_position = false;
    plot_particle_forces = false;
    plot_particle_x_forces = false;
    plot_particle_z_forces = false;
    plot_particle_location = true;
    plot_particle_trajectory = true;
    plot_particle_velocity = true;
    draw_waveguide=true;
    run calculate_particle_trajectory
end
end

% Simulation 5. Graph a particle trajectory for a range of x beam waists.
if (simulation5)
fprintf ('Simulation 3. Graph a particle trajectory for a range of x beam waists.
');
for i=linspace (1,12,24)*1e-6
    run default_parameters;
    x = initial_x;
    z = initial_z;
    x_min_beam_waist = i;
    record_plots = false;
    plot_particle_position = false;
    plot_particle_forces = false;
    plot_particle_x_forces = false;
    plot_particle_z_forces = false;
    plot_particle_location = true;
    plot_particle_trajectory = true;
    plot_particle_velocity = true;
    draw_waveguide=true;
    run calculate_particle_trajectory
end
end

% Simulation 6. Graph a particle trajectory for a range of initial x positions.
if (simulation6)
fprintf ('Simulation 3. Graph a particle trajectory for a range of initial x positions.
');
for i=linspace (0,6,24)*1e-6
    run default_parameters;
    x = i;
    z = initial_z;
    record_plots = false;
    plot_particle_position = false;
    plot_particle_forces = false;
    plot_particle_x_forces = false;
    plot_particle_z_forces = false;
    plot_particle_location = true;
    plot_particle_trajectory = true;
end
end
plot_particle_velocity = true;
draw_waveguide=true;
run calculate_particle_trajectory
end
end

% Simulation 7. Graph a particle trajectory for a range of particle indices.
if (simulation7)
    fprintf ('Simulation 3. Graph a particle trajectory for a range of particle indices.'
    for i=linspace (1.34,1.8,10)
        run default_parameters;
        x = initial_x;
        z = initial_z;
        particle_index
        record_plots = false;
        plot_particle_position = false;
        plot_particle_forces = false;
        plot_particle_x_forces = false;
        plot_particle_z_forces = false;
        plot_particle_location = true;
        plot_particle_trajectory = true;
        plot_particle_velocity = true;
        draw_waveguide=true;
        run calculate_particle_trajectory
    end
end
toc
fprintf ('\n');
beep

calculate_particle_trajectory.m

% calculates particle trajectory

% initialize timestep
timestep = 1;

% run the trajectory simulation for simulation_time (in seconds) with a time resolution of time_resolution (in seconds).
while (true)
    % calculate forces and velocities with particle at x,z
    run equations

    % store results so we can graph them later
    time_vector (timestep) = time;
    x_velocity_vector (timestep) = x_velocity;
    z_velocity_vector (timestep) = z_velocity;
    x_vector (timestep) = x;
    z_vector (timestep) = z;
x_force_vector (timestep) = x_force;
z_force_vector (timestep) = z_force;
x_scattering_force_vector (timestep) = x_scattering_force;
z_scattering_force_vector (timestep) = z_scattering_force;
x_gradient_force_vector (timestep) = x_gradient_force;
z_gradient_force_vector (timestep) = z_gradient_force;

% find next position of particle based on velocity at position z,x
x = x + (x_velocity*time_resolution);
z = z + (z_velocity*time_resolution);

if (include_brownian_motion)
    angle = rand (1)*2*pi;
    x = x + sin (angle)*mean_distance;
    z = z + cos (angle)*mean_distance;
end

time = time + time_resolution;
timestep = timestep + 1;

% Find particle behavior

% Check to see if particle has traveled into the x channel
% When the particle leaves the region where the beams overlap, it will continue in the same direction.
% stop if the particle leaves the region of interest
if (x >= liquid_core_width/2)
    particle_status = 'Particle has entered the X CHANNEL.
    particle_marker = 'b^';
    particle_behavior = 'x';
    run error_checking
    run graphs;
    break;
end

% Check to see if particle has traveled into the z channel
% When the particle leaves the region where the beams overlap, it will continue in the same direction.
% stop if the particle leaves the region of interest
if (z >= solid_core_width/2)
    particle_status = 'Particle has entered the Z CHANNEL.
    particle_marker = 'r>,'
    particle_behavior = 'z';
    run error_checking
    run graphs;
    break;
end

% Check to see if particle has a low enough velocity that it is probably trapped.
% For a 50 mW z beam, the z velocity is about 4 um/s for a particle at z = -liquid_core_width.
% This will assume the particle is trapped if both the x and z velocity are less than 1 nm/s
if (abs (z_velocity) <= 1e-9) && (abs (x_velocity) <= 1e-9)
    particle_status = 'Particle is probably TRAPPED.
    particle_marker = 'gp';
particle behavior = 't';
run graphs;
break;
end

% check to see if the simulation time has run out
if (time >= simulation_time)
    particle_status = 'The simulation has run for the specified
time. Particle Behavior UNKNOWN. You probably need to run the
simulation longer or change the simulation parameters.';
particle_marker = 'kx';
particle_marker_size = 12;
run graphs;
break;
end
end

default_parameters.m

% physical parameters
particle_diameter       = 3e-6;            % [m]
z_beam_power            = 0e-3;           % [W] beam along z towards
right
x_beam_power            = 30e-3;           % [W] beam along x upward
liquid_core_width       = 12e-6;           % [m]
solid_core_width        = 12e-6;           % [m]
liquid_core_height      = 5e-6;            % [m]
solid_liquid_coupling   = 1;               % [fraction of power]
air_solid_coupling      = 1;               % [fraction of power]
particle_index          = 1.6;             % refractive index latex =
1.6
fluid_index             = 1.33;            % refractive index water =
1.33
fluid_viscosity         = 1e-3;            % [Pa*s] = [N*s/m^2] =
[kg/ (m*s)]
z0                      = 0;               % [m] negative is left of
center
x0                      = 0;               % [m] negative is below
center
loss_per_cm             = 100;             % [1/cm] liquid core loss
loss_per_m              = 100*loss_per_cm; % [1/m] = 100* [1/cm]
c                       = 3e8;             % [m/s] speed of light
boltzmann_constant      = 1.3806503e-23;   % m^2 kg s^-2 K^-1
temperature             = 300;             % K
x_radial_offset         = 0;               % [m]
x_axial_distance        = 0;               % [m]
x_light_wavelength      = 532e-9;          % [m]
x_min_beam_waist        = 5.6e-6;          % [m]
z_radial_offset         = 0;               % [m]
z_axial_distance        = 0;               % [m]
z_light_wavelength      = 532e-9;          % [m]
z_min_beam_waist        = 5.6e-6;          % [m]
fx_velocity             = -5e-6;            % [m/s] positive =
downward
fz_velocity             = 30e-6;           % [m/s] positive = left
to right
%% Range values for Cutoff plot
radius_cutoff_radius_min = 0.5e-6; % [m]
radius_cutoff_radius_step = 0.5e-6; % [m]
radius_cutoff_radius_max = 7e-6; % [m]
radius_cutoff_velocity_min = 0.1e-6; % [m/s]
radius_cutoff_velocity_Velstep = 0.1e-6; % [m/s]
radius_cutoff_velocity_stepmax = 10e-6; % [m/s]
radius_cutoff_velocity_max = 100e-6; % [m/s]
radius_cutoff_power_min = 1e-3; % [W]
radius_cutoff_power_step = 0.1e-3; % [W]
radius_cutoff_power_stepmax = 100e-3; % [W]
radius_cutoff_power_max = 403e-3; % [W]

%% simulation parameters
spatial_resolution = 1e-6; % [m]
initial_x = 0e-6; % [m]
initial_z = -12e-6; % [m]
x = initial_x; % [m]
z = initial_z; % [m]
simulation_time = 100; % [s]
time_resolution = 0.1e-3; % [s]
time = 0;
version_number = version();
include_brownian_motion = false;

%% graphing parameters
setenv ('GNUTERM', 'x11')
plot_zero_plane = false;
plot_mode_distribution = false;
plot_x_forces = false;
plot_z_forces = false;
plot_total_forces = false;
view_from_above = false;
draw_waveguide = false;
plot_particle_trajectory = false;
plot_particle_position = false;
plot_particle_forces = false;
plot_particle_x_forces = false;
plot_particle_z_forces = false;
record_plots = false;
plot_particle_location = false;
plot_particle_statistics = false;
plot_power_hitting_particle_field = false;
plot_power_hitting_particle_trajectory = false;
show_colorBar = false;
stop_all_plotting = false;
plot_behavior_boundary_points = false;
plot_velocity_vector_field = false;
plot_z_scattering_forces = false;
plot_x_scattering_forces = false;
plot_z_gradient_forces = false;
plot_x_gradient_forces = false;
plot_total_x_forces = false;
plot_total_z_forces = false;
plot_velocities = false;
plot_z_velocities = false;
plot_x_velocities = false;
plot_particle_behavior_pvd = false;
plot_particle_behavior_pvw = false;
plot_particle_behavior_pvdvw = false;
plot_all_behavior_points = false;
plot_force_field = false;
plot_particle_velocity = false;
plot_particle_velocity_position = false;
plot_rcutoff = false;
plot_rcutoff_velocity = false;
plot_rcutoff_power = false;
plot_flow_velocity_vector_field = false;
plot_flow_velocity_field = false;

clear_figure = true;

particle_status = '';
figure_title = '';
particle_marker = '';
particle_marker_size = 6;
plot_color = 'b';

% initialize data structures
time_vector = 0;
x_velocity_vector = 0;
z_velocity_vector = 0;
x_vector = 0;
z_vector = 0;
x_force_vector = 0;
z_force_vector = 0;
x_scattering_force_vector = 0;
z_scattering_force_vector = 0;
x_gradient_force_vector = 0;
z_gradient_force_vector = 0;
fraction_of_mode_power_x_vector = 0;
fraction_of_mode_power_z_vector = 0;
force_vector = 0;
particle_behavior = '';

% get flow parameters from external simulation
fkx_velocity = fx_velocity; % [m/s]
fkz_velocity = fz_velocity; % [m/s]
load ('flowfields.mat')

equations.m

% find some new values
particle_radius = particle_diameter/2; % [m]
particle_reflectivity = ( (particle_index-fluid_index)/(particle_index+fluid_index))^2; % between 0 and 1
particle_area = pi*particle_radius^2;

% set intial speeds to speeds of flow
run get_flow_velocity
\[ z_0 \text{ velocity} = zvelo; \]
\[ x_0 \text{ velocity} = xvelo; \]
\[ fx \text{ velocity} = x_0 \text{ velocity}; \]
\[ fz \text{ velocity} = z_0 \text{ velocity}; \]

\% define optical beam power coupled into the waveguides
\[ P_0_z = \textit{z Beam power} \times \textit{solid liquid coupling} \times \textit{air solid coupling}; \quad \% \text{ [W]} \]
\[ P_0_x = \textit{x Beam power} \times \textit{solid liquid coupling} \times \textit{air solid coupling}; \quad \% \text{ [W]} \]

\% define optical beam power for the z and x beams.
\[ P_z = (P_0_z \times \text{exp} (-\text{loss per m} \times (z-z_0))); \]
\[ P_x = (P_0_x \times \text{exp} (-\text{loss per m} \times (x-x_0))); \]

\% define scattering forces in the x and z direction
\% Forces applied along direction of beam, along prefix direction
% Reference (http://dx.doi.org/10.1364/JOSAB.23.000897)
x scattering force = \text{dblquad} (@ (\theta_1, \phi) \left( \frac{\text{fluid index}}{2c} \right) \ast \left( \frac{2 \times P_x}{\pi \times (x_{min\ beam\ waist} \times \sqrt{1 + \left(\frac{x_{light\ wavelength} \times x_{axial\ distance}}{\pi \times x_{min\ beam\ waist}^2}\right)^2}} \right) \ast \text{exp} \left( -2 \times \left( \frac{\sqrt{x^2 + \text{particle radius}^2 \times \sin(\theta_1)^2 - 2 \times x \times \text{particle radius} \times \sin(\theta_1 \cos(\phi))}}{x_{min\ beam\ waist} \times \sqrt{1 + \left(\frac{x_{light\ wavelength} \times x_{axial\ distance}}{\pi \times x_{min\ beam\ waist}^2}\right)^2}} \right)^2 \right) \right) \ast \left( 1 + \left( \frac{\sin(\theta_1 - (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right))}{\sin(\theta_1 + (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right)))} \right) \right) \ast \cos(2 \times \theta_1) \right) \ast \left( 1 + \left( \frac{\sin(\theta_1 - (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right))}{\sin(\theta_1 + (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right)))} \right) \right) \ast \cos(2 \times (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right))) \right) \ast \sin(2 \times \theta_1), 1 \ast 10^{-100}, \pi/2, 1 \ast 10^{-100}, \pi/2, 1 \ast 10^{-17}); \]

z scattering force = \text{dblquad} (@ (\theta_1, \phi) \left( \frac{\text{fluid index}}{2c} \right) \ast \left( \frac{2 \times P_z}{\pi \times (z_{min\ beam\ waist} \times \sqrt{1 + \left(\frac{z_{light\ wavelength} \times z_{axial\ distance}}{\pi \times z_{min\ beam\ waist}^2}\right)^2}} \right) \ast \text{exp} \left( -2 \times \left( \frac{\sqrt{x^2 + \text{particle radius}^2 \times \sin(\theta_1)^2 - 2 \times x \times \text{particle radius} \times \sin(\theta_1 \cos(\phi))}}{z_{min\ beam\ waist} \times \sqrt{1 + \left(\frac{z_{light\ wavelength} \times z_{axial\ distance}}{\pi \times z_{min\ beam\ waist}^2}\right)^2}} \right)^2 \right) \right) \ast \left( 1 + \left( \frac{\sin(\theta_1 - (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right))}{\sin(\theta_1 + (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right)))} \right) \right) \ast \cos(2 \times \theta_1) \right) \ast \left( 1 + \left( \frac{\sin(\theta_1 - (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right))}{\sin(\theta_1 + (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right)))} \right) \right) \ast \cos(2 \times (\text{asin} \left( \frac{\text{fluid index}}{\text{particle index} \times \sin(\theta_1)}\right))) \right) \ast \sin(2 \times \theta_1), 1 \ast 10^{-100}, \pi/2, 1 \ast 10^{-100}, \pi/2, 1 \ast 10^{-17});
(fluid_index/particle_index*sin(theta1))).^2 + tan(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / tan(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2 + 2*tan(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / tan(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2)^2).^2).^2).^2)^2).^2)* cos(2.*theta1)*cos2.*theta1) / (1 + (1/2* (sin(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / sin(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2 + tan(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / tan(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2)^2 + 2* (1/2* (sin(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / sin(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2 + tan(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / tan(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2)^2)).^2).^2).^2)^2)).^2)* particle_radius^2.*sin(2.*theta1), 1*10^-100, pi/2, 1*10^-100, 2*pi, 1*10^-17);

% define gradient forces in the x and z direction
% Forces applied along perpendicular to beam, along prefix direction
% Reference (http://dx.doi.org/10.1364/JOSAB.23.000897)
x_gradient_force = dblquad (@ (theta1,phi) (fluid_index/ (2*c).* (Z^2 + pI^2) / (pi^2 * z_min_beam_waist^2 * z_axial_distance) * exp (-2* (sqrt(x^2 + particle_radius^2 * sin(theta1).^2 - 2*x*particle_radius*sin(theta1)*cos(phi))).^2 / (z_min_beam_waist^2 * sqrt(1 + (z_light_wavelength*z_axial_distance / (pi*z_min_beam_waist^2)).^2)).^2)) * ( (1/2* (sin(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / sin(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2 + tan(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / tan(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2)^2 + 2* (1/2* (sin(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / sin(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2 + tan(theta1 - (asin(fluid_index/particle_index*sin(theta1))))).^2 / tan(theta1 + (asin(fluid_index/particle_index*sin(theta1))))).^2)^2)))* particle_radius^2.*sin(2.*theta1), 1*10^-100, pi/2, 1*10^-100, 2*pi, 1*10^-17);
z_gradient_force = dblquad (@ (theta1,phi) (fluid_index/ (2*c).* (2*P_x/ (pi* (x_min_beam_waist*sqrt (1+ (x_light_wavelength*x_axial_distance/ (pi*x_min_beam_waist^2))^2)))*exp (-2* (sqrt (z^2+Particle_radius^2*sin (thet1).^2-2*z*Particle_radius*sin (thet1)*cos (phi))).^2/ (x_min_beam_waist*x_axial_distance/ (pi*x_min_beam_waist^2)))).^2)).* ( (1/2* (sin (theta1- (asin (fluid_index/particle_index*sin (theta1))))).^2/sin (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2+tan (theta1- (asin (fluid_index/particle_index*sin (theta1)))).^2/tan (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2)).*sin (2.*theta1)- (1- (1/2* (sin (theta1- (asin (fluid_index/particle_index*sin (theta1))))).^2/sin (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2+tan (theta1- (asin (fluid_index/particle_index*sin (theta1)))).^2/tan (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2)).*sin (2.*theta1- (1/2* (sin (theta1- (asin (fluid_index/particle_index*sin (theta1))))).^2/sin (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2+tan (theta1- (asin (fluid_index/particle_index*sin (theta1)))).^2/tan (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2)).*sin (2.*theta1)- (1/2* (sin (theta1- (asin (fluid_index/particle_index*sin (theta1))))).^2/sin (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2+tan (theta1- (asin (fluid_index/particle_index*sin (theta1)))).^2/tan (theta1+ (asin (fluid_index/particle_index*sin (theta1)))).^2)).^2)*particle_radius^2.*sin (2*theta1).*cos (phi)),1*10^-100,pi/2,1*10^-100,2*pi,1*10^-17);

% limit the forces to only have values inside liquid-core waveguides.
particle_in_z_beam = ( (x>=-liquid_core_width/2) & (x<=liquid_core_width/2));
particle_in_x_beam = ( (z>=-solid_core_width/2) & (z<=liquid_core_width/2) & (x>=-liquid_core_width/2));
if ~particle_in_z_beam
    z_scattering_force = 0;
    x_gradient_force = 0;
end
if ~particle_in_x_beam
    x_scattering_force = 0;
    z_gradient_force = 0;
end

% calculate total forces in the x and z direction by combining forces
z_force = z_scattering_force + z_gradient_force;
x_force = x_scattering_force + x_gradient_force;

% calculate particle velocity by balancing optical forces with Stokes drag
% drag force = 6*pi*fluid viscosity*particle radius*velocity
% http://www.engineeringtoolbox.com/water-dynamic-kinematic-viscosity-d_596.html
% dynamic viscosity = 1e-3 [Pa*s] at 20 °C (68 °F)
% dynamic viscosity = .8e-3 [Pa*s] at 30 °C (86 °F)
x velocity = x force/
(6*pi*fluid_viscosity*particle_radius)+fx_velocity; % [m/s]
z velocity = z force/
(6*pi*fluid_viscosity*particle_radius)+fz_velocity; % [m/s]

% calculate distance covered by brownian motion
if (include_brownian_motion)
    diffusion_coefficient = boltzmann_constant*temperature/
(6*pi*fluid_viscosity*particle_radius);
    mean_distance         = sqrt
(2*diffusion_coefficient*time_resolution);
end

error_checking.m

% The simulation will not operate correctly if the forces get too
% large, without an increase in the time_resolution of the simulation
% For a 50 mW z beam, the z velocity is about 4 um/s for a particle
% at z = -liquid_core_width.
% If the particle reaches the X channel and the z velocity > x
% velocity, the simulation parameters are probably wrong.
% If the particle reaches the Z channel and the x velocity > z
% velocity, the simulation parameters are probably wrong.
switch (particle_behavior)
    case 'x'
        if (z_velocity > x_velocity)
            particle_status = 'Particle behavior is possibly in
ERROR.
';
            particle_marker = 'kx';
            particle_marker_size = 12;
            particle_behavior = 'e';
        end
    case 'z'
        if (z_velocity < x_velocity)
            particle_status = 'Particle behavior is possibly in
ERROR.
';
            particle_marker = 'kx';
            particle_marker_size = 12;
            particle_behavior = 'e';
        end
end

get_flow_velocity.m

% this creates a flow field using simulated parameters as a starting
point
xv=x*10^6; %convert x position into um
zv=z*10^6; %convert z position into um

% prepare x
Xvalues=unique(flowfieldx(:,2)); %list x value options in imported file
[Xvalmin,XvalInd]=min(abs(Xvalues-xv)); %find which x point is closest to current value
Xval=Xvalues(XvalInd); %define x value to be used
Xind=find(flowfieldx(:,2)==Xval); %find positions that have this x value

% prepare y
Zvalues=flowfieldx(Xind,1); %list z value options in imported file
[Zvalmin,ZvalInd]=min(abs(Zvalues-zv)); %find which z point is closest to current value
Zval=Zvalues(ZvalInd); %define z value to be used
Zind=find(flowfieldx(:,1)==Zval); %find positions that have this z value
ind=mode([Zind,Xind]); %find position that occurs for both our x and z value

% find flow velocity at current position.
zvelo=flowfieldz(ind,3)*fkz_velocity/1e-6+flowfieldx(ind,3)*fkx_velocity/1e-6;
xvelo=flowfieldz(ind,4)*fkz_velocity/1e-6+flowfieldx(ind,4)*fkx_velocity/1e-6;

% Mechanism to override plotting
if (stop_all_plotting)
    break
end

% Plot mode distribution
if (plot_mode_distribution);
    h = figure (1); if (clear_figure); clf; else; hold on; end;
draw_waveguide = false;
subplot (2,1,1); p = plot (x, x_mode_distribution);
subplot (2,1,2); p = plot (z, z_mode_distribution);
end

% Plot Z Scattering Forces in waveguide channel
% z_gradient_force is transposed so that it can be written in the same order as z_scattering_force
if (plot_z_scattering_forces);
    h = figure (2); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
surf (z_vector, x_vector, z_scattering_force_vector','EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel('Z Position [m]'); end
% Plot X Scattering Forces in waveguide channel
% x_scattering_force is transposed so that it can be written in the same order as x_scattering_force
if (plot_x_scattering_forces);
    h = figure (3); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
surf(z_vector, x_vector, x_scattering_force_vector,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Force [N]'); figure_title = 'X Scattering Force [N]'; run graphs_helper;
end

% Plot Z Gradient Forces in waveguide channel
% z_gradient_force is transposed so that it can be written in the same order as z_scattering_force
if (plot_z_gradient_forces);
    h = figure (4); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
surf(z_vector, x_vector, z_gradient_force_vector,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Force [N]'); figure_title = 'Z Gradient Force [N]'; run graphs_helper;
end

% Plot X Gradient Forces in waveguide channel
% x_gradient_force is transposed so that it can be written in the same order as x_scattering_force
if (plot_x_gradient_forces);
    h = figure (5); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
surf(z_vector, x_vector, x_gradient_force_vector,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Force [N]'); figure_title = 'X Gradient Force [N]'; run graphs_helper;
end

% Plot Total Z Forces in waveguide channel
if (plot_total_z_forces);
    h = figure (6); if (clear_figure); clf; else; hold on; end;
    view_from_above = true;
draw_waveguide = true;
surf(z_vector, x_vector, z_force_vector,'EdgeColor','none','FaceColor','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Force [N]'); figure_title = 'Total Z Forces [N]'; run graphs_helper;
end

% Plot Total X Forces in waveguide channel
if (plot_total_x_forces);
    h = figure (7); if (clear_figure); clf; else; hold on; end;
    view_from_above = true;
draw_waveguide = true;
surf(z_vector, x_vector,
x_force_vector,'EdgeColor','none','FaceColor','interp','FaceLighting'
', 'phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]');
zlabel ('Force [N]'); figure_title = 'Total X Forces [N]'; run
graphs_helper;
end

% Plot Z Forcess in waveguide channel
% z gradient force is transposed so that it can be written in the
% same order as z_scattering_force
if (plot_z_forces);
    h = figure (8); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
subplot (1,2,1); surfc (z, x,
z_scattering_force_vector,'EdgeColor','none','FaceColor','interp','Fa
ceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position
[m]'); zlabel ('Force [N]'); figure_title = 'Z Scattering Force
[N]'; run graphs_helper;
subplot (1,2,2); surfc (z, x,
z_gradient_force_vector,'EdgeColor','none','FaceColor','interp','Fac
ceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position
[m]'); zlabel ('Force [N]'); figure_title = 'Z Gradient Force [N]';
run graphs_helper;
end

% Plot X Forces in waveguide channel
% x gradient force is transposed so that it can be written in the
% same order as x_scattering_force
if (plot_x_forces);
    h = figure (9); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
subplot (1,2,1); surf (z_vector, x_vector,
x_scattering_force_vector,'EdgeColor','none','FaceColor','interp','Fa
celighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position
[m]'); zlabel ('Force [N]'); figure_title = 'X Scattering Force
[N]'; run graphs_helper;
subplot (1,2,2); surf (z_vector, x_vector,
x_gradient_force_vector,'EdgeColor','none','FaceColor','interp','Fac
celighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position
[m]'); zlabel ('Force [N]'); figure_title = 'X Gradient Force [N]';
run graphs_helper;
end

% Plot Total Forces in waveguide channel
% z force vector is transposed so that it can be written in the
% same order as z_scattering_force
if (plot_total_forces);
    h = figure (10); if (clear_figure); clf; else; hold on; end;
    view_from_above = true;
draw_waveguide = true;
subplot (1,2,1); surf (z_vector, x_vector,
z_force_vector,'EdgeColor','none','FaceColor','interp','FaceLighting'
', 'phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]');
zlabel ('Force [N]'); figure_title = 'Total Z Forces [N]'; run
graphs_helper;
subplot (1,2,2); surf (z_vector, x_vector,
x_force_vector,'EdgeColor','none','FaceColor','interp','FaceLighting'
', 'phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]');
zlabel ('Force [N]'); figure_title = 'Total X Forces [N]'; run
graphs_helper;
end
%% Plot X and Z velocity in waveguide channel
if (plot_velocities);
    h = figure (11); if (clear_figure); clf; else; hold on; end;
    view_from_above = true;
    draw_waveguide = true;
    subplot (1,2,1); surf (z_vector, x_vector, z_velocity_vector,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Velocity [m/s]'); figure_title = 'Z Velocity [m/s]'; run graphs_helper;
    subplot (1,2,2); surf (z_vector, x_vector, x_velocity_vector,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Velocity [m/s]'); figure_title = 'X Velocity [m/s]'; run graphs_helper;
end

%% Plot X velocity in waveguide channel
if (plot_x_velocities);
    h = figure (12); if (clear_figure); clf; else; hold on; end;
    view_from_above = true;
    draw_waveguide = true;
    surf (z_vector, x_vector, x_velocity_vector,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Velocity [m/s]'); figure_title = 'X Velocity [m/s]'; run graphs_helper;
end

%% Plot Z velocity in waveguide channel
if (plot_z_velocities);
    h = figure (13); if (clear_figure); clf; else; hold on; end;
    view_from_above = true;
    draw_waveguide = true;
    surf (z_vector, x_vector, z_velocity_vector,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Velocity [m/s]'); figure_title = 'Z Velocity [m/s]'; run graphs_helper;
end

%% Plot fraction of mode power hitting particle in waveguide channel
if (plot_power_hitting_particle_field);
    h = figure (14); if (clear_figure); clf; else; hold on; end;
    view_from_above = true;
    draw_waveguide = true;
    subplot (1,2,1); surf (z_vector, x_vector, fraction_of_mode_power_z,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Mode Power Intercepted Z [%]'); figure_title = 'Mode Power Intercepted Z [%]'; run graphs_helper;
    subplot (1,2,2); surf (z_vector, x_vector, fraction_of_mode_power_x,'EdgeColor','none','FaceColor','interp','FaceLighting','phong'); xlabel ('Z Position [m]'); ylabel ('X Position [m]'); zlabel ('Mode Power Intercepted X [%]'); figure_title = 'Mode Power Intercepted X [%]'; run graphs_helper;
end
% Plot velocity vector field in waveguide channel
if (plot_velocity_vector_field);
    h = figure (15); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
show_colorbar = false;
quiver
(z_vector,x_vector,z_velocity_vector',x_velocity_vector','LineWidth',
2);
xlabel ('Z Position [\mum]','fontsize',24); ylabel ('X Position
[\mum]','fontsize',24); zlabel ('Velocity Vector Field','fontsize',24); figure_title = 'Velocity Vector Field'; run
graphs_helper;
set (gca,'YTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'
'YTickLabel','[-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],''XTick','[-
solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],''XTickLabel','[-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],''ZTick','[-sol
d_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],''ZTickLabel','[-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6]);
end

% Plot velocity vector field in waveguide channel
if (plot_velocity_field);
    h = figure (15); if (clear_figure); clf; else; hold on; end;
draw_waveguide = true;
show_colorbar = false;
surf (z_vector,x_vector,sqrt ( (z_velocity_vector').^2+(x_velocity_vector').^2),'EdgeColor','none','FaceColor','interp','Fac
eLighting','phong'); run graphs_helper;
xlabel ('Z Position  [\mum]'); ylabel ('X Position  [\mum]'); zlabel ('Flow Velocity Field  [m/s]'); figure_title = 'Flow Velocity Field'; run graphs_helper;
set (gca,'YTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'
'YTickLabel','[-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],''XTick','[-
solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],''XTickLabel','[-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],''ZTick','[-sol
d_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],''ZTickLabel','[-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6]);
end

% Plot Particle Trajectory
if (plot_particle_trajectory);
    h = figure (16); if (clear_figure); clf; else; hold on; end;
p = plot (z_vector, x_vector);
xlabel ('Z Position  [\mum]','fontsize',24); ylabel ('X Position
[\mum]','fontsize',24); figure_title = 'Particle Trajectory'; run
graphs_helper;
set (p, 'Color', plot_color)
set (gca, 'YTick', [-solid_core_width, -solid_core_width/2, solid_core_width/2, solid_core_width], 'YTickLabel', [-solid_core_width/1e-6, -solid_core_width/2/1e-6, solid_core_width/2/1e-6, solid_core_width/1e-6], 'XTick', [-solid_core_width, -solid_core_width/2, solid_core_width/2, solid_core_width], 'XTickLabel', [-solid_core_width/1e-6, -solid_core_width/2/1e-6, solid_core_width/2/1e-6, solid_core_width/1e-6], 'ZTick', [-solid_core_width, -solid_core_width/2, solid_core_width/2, solid_core_width], 'ZTickLabel', [-solid_core_width/1e-6, -solid_core_width/2/1e-6, solid_core_width/2/1e-6, solid_core_width/1e-6]);
if (draw_waveguide)
draw_waveguide = false;
end;
end

% Plot Particle Position for a particle trajectory
if (plot_particle_position);
h = figure (17); if (clear_figure); clf; else; hold on; end;
draw_waveguide = false;
subplot (1, 2, 1); p = plot (time_vector, x_vector); xlabel ('time [s]'); ylabel ('X Position [m]'); figure_title = 'Particle X Position [m]'; run graphs_helper;
subplot (1, 2, 2); p = plot (time_vector, z_vector); xlabel ('time [s]'); ylabel ('Z Position [m]'); figure_title = 'Particle Z Position [m]'; run graphs_helper;
end

% Plot Particle velocities
if (plot_particle_velocity);
h = figure (27); if (clear_figure); clf; else; hold on; end;
draw_waveguide = false;
subplot (1, 2, 1); p = plot (time_vector, x_velocity_vector); xlabel ('time [s]'); ylabel ('X Velocity [m/s]'); figure_title = 'Particle X Velocity [m/s]'; run graphs_helper;
subplot (1, 2, 2); p = plot (time_vector, z_velocity_vector); xlabel ('time [s]'); ylabel ('Z Velocity [m/s]'); figure_title = 'Particle Z Velocity [m/s]'; run graphs_helper;
end

% Plot Particle velocities
if (plot_particle_velocity_position);
h = figure (28); if (clear_figure); clf; else; hold on; end;
draw_waveguide = false;
subplot (1, 4, 1); p = plot (x_vector, x_velocity_vector); xlabel ('X Position [m]'); ylabel ('X Velocity [m/s]'); figure_title = 'Particle X Velocity [m/s]'; run graphs_helper;
subplot (1, 4, 2); p = plot (z_vector, z_velocity_vector); xlabel ('Z Position [m]'); ylabel ('Z Velocity [m/s]'); figure_title = 'Particle Z Velocity [m/s]'; run graphs_helper;
subplot (1, 4, 3); p = plot (z_vector, x_velocity_vector); xlabel ('Z Position [m]'); ylabel ('X Velocity [m/s]'); figure_title = 'Particle X Velocity [m/s]'; run graphs_helper;
subplot (1, 4, 4); p = plot (x_vector, z_velocity_vector); xlabel ('X Position [m]'); ylabel ('Z Velocity [m/s]'); figure_title = 'Particle Z Velocity [m/s]'; run graphs_helper;
end
% Plot Particle Total Forces for a particle trajectory
if (plot_particle_forces)
    h = figure (18); if (clear_figure); clf; else; hold on; end;
    draw_waveguide = false;
    subplot (1,2,1); p = plot (time_vector, x_force_vector); xlabel ('time [s]'); ylabel ('X Force [N]'); figure_title = 'Particle X Force [N]'; run graphs_helper;
    subplot (1,2,2); p = plot (time_vector, z_force_vector); xlabel ('time [s]'); ylabel ('Z Force [N]'); figure_title = 'Particle Z Force [N]'; run graphs_helper;
end

% Plot Particle Z Forces for a particle trajectory
if (plot_particle_z_forces)
    h = figure (19); if (clear_figure); clf; else; hold on; end;
    draw_waveguide = false;
    subplot (1,2,1); p = plot (time_vector, z_scattering_force_vector); xlabel ('time [s]'); ylabel ('Z Scattering Force [N]'); figure_title = 'Particle Z Scattering Force [N]'; run graphs_helper;
    subplot (1,2,2); p = plot (time_vector, z_gradient_force_vector); xlabel ('time [s]'); ylabel ('Z Gradient Force [N]'); figure_title = 'Particle Z Gradient Force [N]'; run graphs_helper;
end

% Plot Particle X Forces for a particle trajectory
if (plot_particle_x_forces)
    h = figure (20); if (clear_figure); clf; else; hold on; end;
    draw_waveguide = false;
    subplot (1,2,1); p = plot (time_vector, x_scattering_force_vector); xlabel ('time [s]'); ylabel ('X Scattering Force [N]'); figure_title = 'Particle X Scattering Force [N]'; run graphs_helper;
    subplot (1,2,2); p = plot (time_vector, x_gradient_force_vector); xlabel ('time [s]'); ylabel ('X Gradient Force [N]'); figure_title = 'Particle X Gradient Force [N]'; run graphs_helper;
end

% Plot fraction of mode power hitting particle for a particle trajectory
if (plot_power_hitting_particle_trajectory)
    h = figure (21); if (clear_figure); clf; else; hold on; end;
    draw_waveguide = false;
    subplot (1,2,1); p = plot (time_vector, fraction_of_mode_power_x_vector); xlabel ('time [s]'); ylabel ('Mode Power Intercepted X'); figure_title = 'Mode Power Intercepted X'; run graphs_helper;
    subplot (1,2,2); p = plot (time_vector, fraction_of_mode_power_z_vector); xlabel ('time [s]'); ylabel ('Mode Power Intercepted Z'); figure_title = 'Mode Power Intercepted Z'; run graphs_helper;
end

% Plot particle behavior while varying beam power ratio and particle diameter
if (plot_particle_behavior_pvd)
    h = figure (22); if (clear_figure); clf; else; hold on; end;
    % generate ordered polygon
reordered_z = []; reordered_t = []; reordered_x = []; skipped_z = []; skipped_t = []; skipped_x = []; top_row_z = []; top_row_t = []; top_row_x = [];

% For X behavior, grab first data point of each row until we get to end of list
% add top points in reverse order
% add skipped points in reverse order
for reorder_index = [1:length (z_behavior_points)]
  if (z_behavior_points (reorder_index,2) == particle_diameter_range (end))
    top_row_z = [top_row_z; z_behavior_points (reorder_index,:)];
  else
    reordered_z = [reordered_z; z_behavior_points (reorder_index,:)];
  end
end
reordered_z = [reordered_z; flipud (top_row_z)]; reordered_z = [reordered_z; flipud (skipped_z)];

% For T behavior, grab every other data point of each row until we get to end of list
% add top points in reverse order
% add skipped points in reverse order
for reorder_index = [1:length (t_behavior_points)]
  if (t_behavior_points (reorder_index,2) == particle_diameter_range (end))
    top_row_t = [top_row_t; t_behavior_points (reorder_index,:)];
  elseif ~mod (reorder_index,2)
    reordered_t = [reordered_t; t_behavior_points (reorder_index,:)];
  else
    skipped_t = [skipped_t; t_behavior_points (reorder_index,:)];
  end
end
reordered_t = [reordered_t; flipud (top_row_t)]; reordered_t = [reordered_t; flipud (skipped_t)];

% For X behavior, grab last data point of each row until we get to end of list
% add top points in reverse order
% add skipped points in reverse order
for reorder_index = [1:length (x_behavior_points)]
  if (x_behavior_points (reorder_index,2) == particle_diameter_range (end))
    top_row_x = [top_row_x; x_behavior_points (reorder_index,:)];
  end
end
elseif x_behavior_points (reorder_index,1) == beam_power_ratio_range (end)
    reordered_x = [reordered_x; x_behavior_points (reorder_index,:)];
else
    skipped_x = [skipped_x; x_behavior_points (reorder_index,:)];
end
reordered_x = [reordered_x; flipud (top_row_x)];
reordered_x = [reordered_x; flipud (skipped_x)];
hold on
fill (reordered_z (:,1), reordered_z (:,2)*1e6, 'r');
fill (reordered_t (:,1), reordered_t (:,2)*1e6, 'g');
fill (reordered_x (:,1), reordered_x (:,2)*1e6, 'b');
hold off
xlabel ('Beam power ratio X/Z'); ylabel ('Particle diameter [um]');
title ('Particle Behavior Regions');
legend ('Z channel', 'Trapped', 'X channel')

eend

% Plot particle behavior while varying beam power ratio and solid core width
if (plot_particle_behavior_pvw)
    h = figure (23); if (clear_figure); clf; else; hold on; end;
    % generate ordered polygon
    reordered_z = [];
    reordered_t = [];
    reordered_x = [];
    skipped_z = [];
    skipped_t = [];
    skipped_x = [];
    top_row_z = [];
    top_row_t = [];
    top_row_x = [];

    % For X behavior, grab first data point of each row until we get to end of list
    % add top points in reverse order
    % add skipped points in reverse order
    for reorder_index = [1:length (z_behavior_points)]
        if (z_behavior_points (reorder_index,2) == solid_core_width_range (end))
            top_row_z = [top_row_z; z_behavior_points (reorder_index,:)];
        elseif z_behavior_points (reorder_index,1) == beam_power_ratio_range (1)
            reordered_z = [reordered_z; z_behavior_points (reorder_index,:)];
        else
            skipped_z = [skipped_z; z_behavior_points (reorder_index,:)];
        end
reordered_z = [reordered_z; flipud (top_row_z)];
reordered_z = [reordered_z; flipud (skipped_z)];

% For T behavior, grab every other data point of each row until we get to end of list
% add top points in reverse order
% add skipped points in reverse order
for reorder_index = [1:length (t_behavior_points)]
    if (t_behavior_points (reorder_index,2) ==
    solid_core_width_range (end))
        top_row_t = [top_row_t; t_behavior_points
    (reorder_index,:)];
    elseif ~mod (reorder_index,2)
        reordered_t = [reordered_t; t_behavior_points
    (reorder_index,:)];
    else
        skipped_t = [skipped_t; t_behavior_points
    (reorder_index,:)];
    end
end
reordered_t = [reordered_t; flipud (top_row_t)];
reordered_t = [reordered_t; flipud (skipped_t)];

% For X behavior, grab last data point of each row until we get to end of list
% add top points in reverse order
% add skipped points in reverse order
for reorder_index = [1:length (x_behavior_points)]
    if (x_behavior_points (reorder_index,2) ==
    solid_core_width_range (end))
        top_row_x = [top_row_x; x_behavior_points
    (reorder_index,:)];
    elseif x_behavior_points (reorder_index,1) ==
    beam_power_ratio_range (end)
        reordered_x = [reordered_x; x_behavior_points
    (reorder_index,:)];
    else
        skipped_x = [skipped_x; x_behavior_points
    (reorder_index,:)];
    end
end
reordered_x = [reordered_x; flipud (top_row_x)];
reordered_x = [reordered_x; flipud (skipped_x)];

hold on
if ~isempty (reordered_z) fill (reordered_z (:,1), reordered_z
(:,2)*1e6, 'r'); end
if ~isempty (reordered_t) fill (reordered_t (:,1), reordered_t
(:,2)*1e6, 'g'); end
if ~isempty (reordered_x) fill (reordered_x (:,1), reordered_x
(:,2)*1e6, 'b'); end
hold off
xlabel ('Beam power ratio X/Z'); ylabel ('X channel width
[um]'); title ('Particle Behavior Regions');
legend ('Z channel', 'Trapped', 'X channel')
end
% Plot Behavior Boundary Points
if (plot_behavior_boundary_points)
    h = figure (24); if (clear_figure); clf; else; hold on; end;
    hold on
    if ~isempty (z_behavior_points) plot (z_behavior_points (:,1),
                                       z_behavior_points (:,2)*1e6, 'r>'); end
    if ~isempty (t_behavior_points) plot (t_behavior_points (:,1),
                                       t_behavior_points (:,2)*1e6, 'g*'); end
    if ~isempty (x_behavior_points) plot (x_behavior_points (:,1),
                                       x_behavior_points (:,2)*1e6, 'b^'); end
    plot (all_behavior_points (:,1)*1e3, all_behavior_points
          (:,2)*1e6, 'k.'); end
    if ~isempty (other_behavior_points) plot (other_behavior_points
                                             (:,1), other_behavior_points (:,2)*1e6, 'kx'); end
    hold off
    xlabel ('Beam power ratio X/Z'); ylabel ('Particle diameter
[um]'); title ('Particle Behavior Regions');
end

% Plot Behavior Boundary Points
if (plot_all_behavior_points)
    h = figure (25); if (clear_figure); clf; else; hold on; end;
    hold on
    if ~isempty (all_behavior_points)
        for behavior_index = [1:length (all_behavior_points)]
            plot (all_behavior_points (behavior_index,1),
               all_behavior_points (behavior_index,2)*1e6, particle_markers
               (behavior_index,:));
        end
    end
    hold off
    xlabel ('Beam power ratio X/Z'); ylabel ('Particle diameter
[um]'); title ('Particle Behavior Regions');
end

% Plot 3D Behavior Points
if (plot_particle_behavior_pvdvw)
    h = figure (26); if (clear_figure); clf; else; hold on; end;
    hold on
    if ~isempty (all_behavior_points)
        for behavior_index = [1:length (all_behavior_points)]
            plot3 (all_behavior_points (behavior_index,1),
               all_behavior_points (behavior_index,2)*1e6, all Behavior points
               (behavior_index,3)*1e6, particle_markers (behavior_index,:));
        end
    end
    hold off
    xlabel ('Beam power ratio X/Z'); ylabel ('Particle diameter
[um]'); zlabel ('X Beam Width [um]'); title ('Particle Behavior
Regions');
    h = figure (29); if (clear_figure); clf; else; hold on; end;
    hold on

    color = 'r'
    p = all_z_behavior_points .* (ones (1,length
                                        (all_z_behavior_points))* [1 1e6 1e6])
    run TestMyCrust
\[ p = \text{all}_t\text{\_behavior\_points} \times (\text{ones}(1, \text{length} \text{all}_t\text{\_behavior\_points}))' \times [1 \ 1e6 \ 1e6] \]

color = 'g'
run TestMyCrust

\[ p = \text{all}_x\text{\_behavior\_points} \times (\text{ones}(1, \text{length} \text{all}_x\text{\_behavior\_points}))' \times [1 \ 1e6 \ 1e6] \]

color = 'b'
run TestMyCrust

hold off
xlabel ('Beam power ratio X/Z'); ylabel ('Particle diameter [um]'); zlabel ('X Beam Width [um]'); title ('Particle Behavior Regions'); end

% Plot Particle Statistics
if (plot_particle_statistics)
    fprintf ('Initial and final values of particle parameters
')
    fprintf ('time\ttime\[s\]time\[s\]\n',
        time_vector (1), time_vector (end));
    fprintf ('particle diameter\[um\]\n',
        particle_diameter*1e6);
    fprintf ('z beam power\[\text{mW}\]\n',
        z_beam_power*1e3);
    fprintf ('z force\[\text{fN}\]\text{fN}\]\n',
        z_force_vector (1)*1e16, z_force_vector (end)*1e16);
    fprintf ('z scattering force\[\text{fN}\]\text{fN}\]\n',
        z_scattering_force_vector (1)*1e16, z_scattering_force_vector (end)*1e16);
    fprintf ('z gradient force\[\text{fN}\]\text{fN}\]\n',
        z_gradient_force_vector (1)*1e16, z_gradient_force_vector (end)*1e16);
    fprintf ('z velocity\[\text{um/s}\]\text{um/s}\]\n',
        z_velocity_vector (1)*1e6, z_velocity_vector (end)*1e6);
    fprintf ('z mode fraction\[\%\]\text{fN}\]\n',
        fraction_of_mode_power_z_vector (1)*1e2, fraction_of_mode_power_z_vector (end)*1e2);
    fprintf ('x beam power\[\text{mW}\]\n',
        x_beam_power*1e3);
    fprintf ('x force\[\text{fN}\]\text{fN}\]\n',
        x_force_vector (1)*1e16, x_force_vector (end)*1e16);
    fprintf ('x scattering force\[\text{fN}\]\text{fN}\]\n',
        x_scattering_force_vector (1)*1e16, x_scattering_force_vector (end)*1e16);
    fprintf ('x gradient force\[\text{fN}\]\text{fN}\]\n',
        x_gradient_force_vector (1)*1e16, x_gradient_force_vector (end)*1e16);
    fprintf ('x velocity\[\text{um/s}\]\text{um/s}\]\n',
        x_velocity_vector (1)*1e6, x_velocity_vector (end)*1e6);
    fprintf ('x mode fraction\[\%\]\text{fN}\]\n',
        fraction_of_mode_power_x_vector (1)*1e2, fraction_of_mode_power_x_vector (end)*1e2);
    fprintf (particle_status);
    fprintf ('
'); -
end
if (plot_force_field);
    h = figure (28); if (clear_figure); clf; else; hold on; end;
    surf (z_vector, x_vector, force_vector', 'EdgeColor', 'none', 'FaceColor', 'interp', 'FaceLighting', 'phong');
    xlabel ('Z Position [\mum]','fontsize',24); ylabel ('X Position [\mum]', 'fontsize',24); zlabel ('abs (X Force) + abs (Z Force) [N]', 'fontsize',24); figure_title = 'abs (X Force) + abs (Z Force) [N]'; run graphs_helper;
    set (gca,'YTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'YTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],'XTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'XTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6], 'ZTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'ZTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6]);
end

% Plot velocity vector field in waveguide channel
if (plot_flow_velocity_vector_field);
    h = figure (29); if (clear_figure); clf; else; hold on; end;
    draw_waveguide = true;
    show_colorbar = false;
    quiver (z_vector, x_vector, fz_velocity_vector', fx_velocity_vector'); run graphs_helper;
    xlabel ('Z Position [\mum]'); ylabel ('X Position [\mum]');
    zlabel ('Flow Velocity Vector Field [m/s]'); figure_title = 'Flow Velocity Vector Field'; run graphs_helper;
    set (gca,'YTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'YTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],'XTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'XTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6], 'ZTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'ZTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6]);
end

% Plot velocity vector field in waveguide channel
if (plot_flow_velocity_field);
    h = figure (30); if (clear_figure); clf; else; hold on; end;
    draw_waveguide = true;
    show_colorbar = false;
    surf (z_vector, x_vector, fxfz_velocity_magnitude', 'EdgeColor', 'none', 'FaceColor', 'interp', 'FaceLighting', 'phong'); run graphs_helper;
    xlabel ('Z Position [\mum]'); ylabel ('X Position [\mum]');
    zlabel ('Flow Velocity Field [m/s]'); figure_title = 'Flow Velocity Field [m/s]'; run graphs_helper;
    set (gca,'YTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'YTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],'XTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'XTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6], 'ZTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'ZTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6]);
end

% Plot velocity vector field in waveguide channel
if (plot_flow_velocity_field);
    h = figure (30); if (clear_figure); clf; else; hold on; end;
    draw_waveguide = true;
    show_colorbar = false;
    surf (z_vector, x_vector, fxfz_velocity_magnitude', 'EdgeColor', 'none', 'FaceColor', 'interp', 'FaceLighting', 'phong'); run graphs_helper;
    xlabel ('Z Position [\mum]'); ylabel ('X Position [\mum]');
    zlabel ('Flow Velocity Field [m/s]'); figure_title = 'Flow Velocity Field [m/s]'; run graphs_helper;
    set (gca,'YTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'YTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6],'XTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'XTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6], 'ZTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width],'ZTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6]);
end
Field'; run graphs_helper;
    set (gca,'YTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width], 'YTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6], 'XTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width], 'XTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6], 'ZTick', [-solid_core_width,-solid_core_width/2,solid_core_width/2,solid_core_width], 'ZTickLabel', [-solid_core_width/1e-6,-solid_core_width/2/1e-6,solid_core_width/2/1e-6,solid_core_width/1e-6]);
end

%Plot R cutoff
if (plot_rcutoff);
    h = figure (31); if (clear_figure); clf; else; hold on; end;
    plot (R_cutoff_ratio, R_cutoff); ylabel ('Cutoff Radius [m]'); xlabel ('V_y/P [m/Ws]'); figure_title = 'Cutoff Radius'; run graphs_helper;
end

%Plot R cutoff power
if (plot_rcutoff_power);
    h = figure (32); if (clear_figure); clf; else; hold on; end;
    plot (R_cutoff_ratio, R_cutoff_power); ylabel ('Cutoff Radius [m]'); xlabel ('P [W]'); figure_title = 'Cutoff Radius'; run graphs_helper;
end

%Plot R cutoff velocity
if (plot_rcutoff_velocity);
    h = figure (33); if (clear_figure); clf; else; hold on; end;
    plot (R_cutoff_ratio, R_cutoff_velocity); ylabel ('Cutoff Radius [m]'); xlabel ('V_y [m/s]'); figure_title = 'Cutoff Radius'; run graphs_helper;
end

graphs_helper.m

%Graphs Helper
title (figure_title);
set (h, 'name', figure_title);
if (show_colorbar)
    colorbar ('EastOutside');
end
if (view_from_above); view (2); end
if (draw waveguide);
hold on;
% horizontal lines
line ( [solid_core_width, -solid_core_width/2],
[liquid_core_width/2, liquid_core_width/2], [1000,1000]);
line ( [solid_core_width, -solid_core_width/2],
[-liquid_core_width/2, -liquid_core_width/2], [1000,1000]);
line ( [solid_core_width/2, solid_core_width],
[liquid_core_width/2, liquid_core_width/2], [1000,1000]);
line ( [solid_core_width/2, solid_core_width],
[-liquid_core_width/2, -liquid_core_width/2], [1000,1000]);
line ( [-solid_core_width, solid_core_width],
[-liquid_core_width/2, -liquid_core_width/2], [1000,1000]);
% vertical lines
line ( [-solid_core_width/2, -solid_core_width/2],  
[-liquid_core_width/2, -liquid_core_width], [1000,1000]);
line ( [solid_core_width/2, solid_core_width/2],  
[-liquid_core_width/2, -liquid_core_width], [1000,1000]);
line ( [-solid_core_width/2, -solid_core_width/2],
[liquid_core_width/2, liquid_core_width], [1000,1000]);
line ( [solid_core_width/2, solid_core_width/2],
[liquid_core_width/2, liquid_core_width], [1000,1000]);

% Plot Particle Location
if (plot_particle_location);
plot (z, x, particle_marker, 'markersize',
particle_marker_size);
end
% this axis definition is meant to keep a standard view even if
we vary the x waveguide width. The standard waveguide width is 12 um.
axis ( [-12e-6 12e-6 -12e-6 12e-6]);
hold off;
if (record_plots);
filename=sprintf ('%s_%f.png', figure_title, i);
print (filename);
end
if (plot_zero_plane);
% define zero plane
zero_plane = zeros (length (x), length (z));
hold on;
if (version_number (1) ~= '3')
surf (z,x,zero_plane,'EdgeColor','none','FaceColor','w','FaceLighting','ph ong');
end
hold off;
end
clear
clc
close all
fprintf ('Finds Power Cutoff Value.
');
steper=1;
run default_parameters
R_cutoff=radius_cutoff_radius_min:radius_cutoff_radius_step:radius_cu toff_radius_max;

125
for diameter=R_cutoff
diameter
power_start=radius_cutoff_power_max;
power_finish=radius_cutoff_power_min;
power_step=radius_cutoff_power_Stepmax;
loopstill=1;
while (loopstill)
zprevious=initial_z;
xprevious=initial_x;
for power=power_start:-power_step:power_finish
run default_parameters
particle_diameter=diameter;
x Beam_power= power;
run calculate_particle_trajectory
if x>=liquid_core_width/2
if zprevious>=solid_core_width/2
power_step=power_step/10
if power_step<radius_cutoff_power_step
loopstill=0;
R_cutoff_ratio (step) = fz_velocity/power;
R_cutoff_power (step) = power;
R_cutoff_velocity (step) = fz_velocity;
step=step+1;
end
break;
end
else if z>=solid_core_width/2
if xprevious>=liquid_core_width/2
power_step=power_step/10
if power_step<radius_cutoff_power_step
loopstill=0;
R_cutoff_ratio (step) = fz_velocity/power;
R_cutoff_power (step) = power;
R_cutoff_velocity (step) = fz_velocity;
step=step+1;
end
break;
end
end
zprevious=z;
xprevious=x;
if power<radius_cutoff_power_min+radius_cutoff_power_step
fprintf ('This one did not work.
');
loopstill=0;
R_cutoff_ratio (step) = 0;
R_cutoff_power (step) = 0;
R_cutoff_velocity (step) = 0;
step=step+1;
end
end
end

plot rcutoff = false;
plot_rcutoff_power = true;
plot_rcutoff_velocity = false;
run graphs

power_cutoff.m

clear
cic
close all
fprintf ('Finds Power Cutoff Value.\n');

steper=1;
run default_parameters

R_cutoff=radius_cutoff_radius_min:radius_cutoff_radius_step:radius_cutoff_radius_max;

for diameter=R_cutoff
diameter
  power_start=radius_cutoff_power_max;
  power_finish=radius_cutoff_power_min;
  power_step=radius_cutoff_power_STEPmax;
  loopstill=1;
  while (loopstill)
    z_previous=initial_z;
    x_previous=initial_x;
    for power=power_start:-power_step:power_finish
      run default_parameters
      particle_diameter=diameter;
      x_beam_power=power;
      run calculate_particle_trajectory
      if x>=liquid_core_width/2
        if z_previous>=solid_core_width/2
          power_start=power+power_step;
          power_step=power_step/10
          power_finish=power;
          if power_step<radius_cutoff_power_step
            loopstill=0;
            power
            R_cutoff_ratio (steper) = fz_velocity/power;
            R_cutoff_power (steper) = power;
            R_cutoff_velocity (steper) = fz_velocity;
            steper=steper+1;
            end
            break;
          end
        else if z>solid_core_width/2
          if x_previous>=liquid_core_width/2
            power_start=power+power_step;
            power_step=power_step/10
            power_finish=power;
            if power_step<radius_cutoff_power_step
              loopstill=0;
              power
              R_cutoff_ratio (steper) = fz_velocity/power;
              R_cutoff_power (steper) = power;
              R_cutoff_velocity (steper) = fz_velocity;
              steper=steper+1;
              end
              break;
            end
          else
            break;
          end
        end
      end
    end
  end
end
R_cutoff_ratio (steper) = fz_velocity/power;
R_cutoff_power (steper) = power;
R_cutoff_velocity (steper) = fz_velocity;
steper=steper+1;
end
break;
end
end
zprevious=z;
xprevious=x;
if power<radius_cutoff_power_min+radius_cutoff_power_step
fprintf ('This one did not work.
');
loopstill=0;
R_cutoff_ratio (steper) = 0;
R_cutoff_power (steper) = 0;
R_cutoff_velocity (steper) = 0;
steper=steper+1;
end
end
end
plot_rcutoff = false;
plot_rcutoff_power = true;
plot_rcutoff_velocity = false;
run graphs

velocity_cutoff.m

clear
clc
close all
fprintf ('Finds Velocity Cutoff Value.\n');
steper=1;
run default_parameters
R_cutoff=radius_cutoff_radius_min:radius_cutoff_radius_step:radius_cutoff_radius_max;
for diameter=R_cutoff
diameter
zprevious=initial_z;
xprevious=initial_x;
velocity_start=radius_cutoff_velocity_min;
velocity_step=radius_cutoff_velocity_stepmax;
velocity_finish=radius_cutoff_velocity_max;
loopstill=1;
while (loopstill)
zprevious=initial_z;
xprevious=initial_x;
for velocity=velocity_start:velocity_step:velocity_finish
run default_parameters
particle diameter=diameter;
fz_velocity= velocity;
run calculate_particle_trajectory
if x>=liquid_core_width/2
    if zprevIeous>=solid_core_width/2
        velocity_start=velocity-velocity_step;
        velocity_step=velocity_step/10;
        velocity_finish=velocity;
        if velocity_step<radius_cutoff_velocity_step
            loopstill=0;
            velocity
            R_cutoff_ratio (steper) =
            velocity/x_beam_power;
            R_cutoff_velocity (steper) = velocity;
            steper=steper+1;
        end
        break;
    end
elseif z>=solid_core_width/2
    if xprevIeous>=liquid_core_width/2
        velocity_start=velocity-velocity_step;
        velocity_step=velocity_step/10;
        velocity_finish=velocity;
        if velocity_step<radius_cutoff_velocity_step
            loopstill=0;
            velocity
            R_cutoff_ratio (steper) =
            velocity/x_beam_power;
            R_cutoff_velocity (steper) = velocity;
            steper=steper+1;
        end
        break;
    end
    zprevIeous=z;
    xprevIeous=x;
if velocity>radius_cutoff_velocity_max-
radius_cutoff_velocity_step
    fprintf ('Not Found.
');
end
end
end
plot_rcutoff = false;
plot_rcutoff_velocity=true;
plot_rcutoff_power=false;
run graphs
% initialize timestep
timestepx = 1;
timestepz = 1;

% The tic and toc commands measure the amount of time the simulation takes.
% This is useful for simulations that run for a long time, so you can get a sense of when to check on them.
tic

fprintf ('Create force, velocity, and flow fields.\n');
run default_parameters;

for x=-liquid_core_width:spatial_resolution:liquid_core_width
    for z=-solid_core_width :spatial_resolution:solid_core_width
        run equations

        z_scattering_force_vector (timestepz,timestepx) =
        x_scattering_force_vector (timestepz,timestepx) =
        x_scattering_force;
        z_scattering_force_vector (timestepz,timestepx) =
        z_gradient_force_vector (timestepz,timestepx) =
        x_gradient_force_vector (timestepz,timestepx) =
        x_force_vector (timestepz,timestepx) =
        fx_velocity_vector (timestepz,timestepx) =
        fx_velocity;
        fz_velocity_vector (timestepz,timestepx) =
        fz_velocity;
        fxz_velocity_magnitude (timestepz,timestepx) = sqrt
        (fz_velocity^2+fx_velocity^2);
        x_velocity_vector (timestepz,timestepx) =
        x_velocity;
        z_velocity_vector (timestepz,timestepx) =
        z_velocity;
        force_vector (timestepz,timestepx) = abs
        (x_force)+abs (z_force);
        x_vector (timestepz) = x;
        z_vector (timestepz) = z;

        timestepz = timestepz + 1;
    end
end
timestepx=timestepx+1;
timestepz = 1;
end

% plot forces
plot_z_scattering_forces = true;
plot_x_scattering_forces = true;
plot_z_gradient_forces = true;
plot_x_gradient_forces = true;
plot_total_x_forces = true;
plot_total_z_forces = true;
plot_force_field = true;
% plot velocities
plot_x_velocities = true;
plot_z_velocities = true;
plot_velocity_vector_field = true;
plot_flow_velocity_vector_field = true;
plot_flow_velocity_field = true;
show_colorbar = true;
view_from_above = true;
plot_zero_plane = false;

run graphs;
toc
fprintf ('\n');
beep

Calculate_flow_Field.m

clear
clc
close all
% This program does flow velocity fields

% initialize timestep
timestepx = 1;
timestepz = 1;

% The tic and toc commands measure the amount of time the simulation takes.
% This is useful for simulations that run for a long time, so you can get a sense of when to check on them.
tic

toc
fprintf ('Create a velocity field.\n');
run default_parameters;

for x=-liquid_core_width:spatial_resolution:liquid_core_width
    for z=-solid_core_width :spatial_resolution:solid_core_width
        run equations
        fx_velocity_vector (timestepz,timestepx) = fx_velocity;
        fz_velocity_vector (timestepz,timestepx) = fz_velocity;
        fxfz_velocity_magnitude (timestepz,timestepx) = sqrt (fz_velocity^2+fx_velocity^2);
        x_vector (timestepx) = x;
        z_vector (timestepz) = z;

        timestepz = timestepz + 1;
    end
end
timestepx=timestepx+1;
timestepz = 1;
end
plot_flow_velocity_vector_field = true;
plot_flow_velocity_field = true;

run graphs;
toc
fprintf ('\n');
beep
### 10.2 Particle Tracking Software

**Tracking software for large movement**

```matlab
function values=ParticleTrackFunction(filename,Back,rotate,crop,Guide,frame,frameend)

%Plots trajectories of particles in waveguide.
%Code by Kaelyn Leake
% for a single frame:
ParticleTrackFunction(filename,Back,rotate,crop,Guide,frame)
% for all frames:
ParticleTrackFunction(filename,Back,rotate,crop,Guide)
% for range of frames:
ParticleTrackFunction(filename,Back,rotate,crop,Guide,framestart,frameend)

% filename should be in form of string it excepts tifs ex. 'C:/Desktop/130728'
% Back is the frame of the background, ideally a frame without any beads but any will work ex. 1
% rotate is 1 if you want to rotate the video else 0 --> orientation
% expected __ or __
% crop is 0 if you don't want to crop otherwise it's [xmincrop,ymincrop,widthcrop,heightcrop]
% Guide is [topy,bottomy,leftx,rightx] an ex: [511,567,206,260] for intersection....[topy,bottomy] an ex: [511,567] for straight guide...0 if % whole image

clc
close all

%Define Waveguide
if length(Guide)==4 %if intersection
topy=Guide(1);
bottomy=Guide(2);
leftx=Guide(3);
rightx=Guide(4);
elseif length(Guide)==2 %if straight channel
topy=Guide(1);
bottomy=Guide(2);
end

%Crop and then define the waveguide
if length(crop)==4
xmincrop=crop(1);
ymincrop=crop(2);
widthcrop=crop(3);
heightcrop=crop(4);
if length(Guide)==4
```

---

133
topy=topy-ymincrop;
bottomy=bottomy-ymincrop;
leftx=leftx-xmincrop;
rightx=rightx-xmincrop;
elseif length(Guide)==2
    topy=topy-ymincrop;
bottomy=bottomy-ymincrop;
leftx=xmincrop;
rightx=xmincrop+widthcrop;
end
end

%display file info on the m-file
display('The Program is Now Running!')
slash = strfind(filename, '/');
filenum=str2double(filename(slash(end)+1:end));
fname=sprintf('%s.tif',filename);

%get File info and open file
info = imfinfo(fname);
B= imread(fname, Back, 'Info', info);

%check to see if any rotation is necessary
if rotate==1
    B=flipud(rot90(B));
    B=flipud(B);
end

%do cropping if required
if length(crop)==4
    B=imcrop(B,[xmincrop,ymincrop,widthcrop,heightcrop]);
end

%display the background
[height,width]=size(B);
figure(1)
colormap('gray'), imagesc(B);
set(gcf,'name','Background','numbertitle','off')

%create boundaries for none given values
if length(crop)~=4
    if length(Guide)~=4
        leftx=1;
        rightx=width;
        if length(Guide)~=2
            topy=height;
            bottomy=1;
        end
    end
end
%determine range of file
if nargin==7
    startframe=frame;
    stopframe=frameend;
    framerange=startframe:stopframe;
elseif nargin==5
    startframe=1;
    stopframe=numel(info);
    framerange=startframe:stopframe;
else
    framerange=frame;
end
if nargin~==6
    newimage=zeros(height,width,stopframe); %to much for some
    computers may need to comment this and line 169
end

%check to see if done
isitdone=0;
while isitdone==0;
    %clear values before running
    clear centroid centroidmult vals pkn

    %open each tiff frame
    for frame=framerange
        %This function takes an image and adjusts contrasts and
        identifies objects
        A= imread(fname, frame, 'Info', info);
        if rotate==1 %rotate
            A=flipud(rot90(A));
            A=flipud(A);
        end
        if length(crop)==4 %crop
            A=imcrop(A,[xmincrop,ymincrop,widthcrop,heightcrop]);
        end
        E=B-A; %background substract image
        I3=imadjust(E); %constrasts background substracted image

        %black out non-channel area-blacks out areas of image to
        speed up run time
        if length(Guide)==4
            I3(1:topy,1:leftx)=0;
            I3(1:topy,rightx:width)=0;
            I3(bottomy:height,1:width)=0;
        elseif length(Guide)==2
            I3(1:topy,1:width)=0;
            I3(bottomy:height,1:width)=0;
        end

        % level sets threshold for pixels in image to be either
        black or white
% ie when level=0.6 then the darkest 60% of pixels become black and the rest(40%) become white
level = 0.7; %was set to 0.7 but changed for low contrast
plot
bwa = im2bw(I3,level); %converts to black and white
bwb = bwareaopen(bwa, 15); %2nd parameter(15) sets the minimum pixel size to keep in the bw image
%the minimum pixel size is the particle area. ~15 pixels for 0.5um bead
bw = imfill(bwb,'holes'); %fills pixel area selected so that no holes in "particle"
[Bs,L] = bwboundaries(bw,'noholes'); %locates "particle"
stats = regionprops(L,'Area','Centroid','MinorAxisLength','Extrema','MajorAxisLength'); %finds stats on all particles in image

%display images for single frame
if nargin==6
    figure(2)
    colormap('gray'), imagesc(A);
    set(gcf,'name','Original Figure','numbertitle','off')
    figure(3)
    colormap('gray'), imagesc(I3);
    set(gcf,'name','black out non-channel','numbertitle','off')
    figure(4)
    colormap('gray'), imagesc(bwa);
    set(gcf,'name','Convert to black and white','numbertitle','off')
    figure(5)
    colormap('gray'), imagesc(bwb);
    set(gcf,'name','Remove small objects','numbertitle','off')
    figure(6)
    colormap('gray'), imagesc(bw);
    set(gcf,'name','Fill holes in particles','numbertitle','off')
else
    clear centroid centroidmult
    newimage(:,:,frame-startframe+1)=bw;
end

graindata = regionprops(bw, 'all');
statsarea = cat(1, graindata.Area);

threshold = 0.4; %sets max ratio 4*pi*Area/perimeter^2

% Makes sure not background frame....
if max(statsarea)<20000
%The function determines if a group of white pixels is a circle, if a group of white pixels is too large to be a particle, if 2 particles are stuck together.

for k = 1:length(Bs)
    % obtain (X,Y) boundary coordinates corresponding to label 'k'
    boundary = Bs{k};

    % compute a simple estimate of the object's perimeter
    delta_sq = diff(boundary).^2;
    perimeter = sum(sqrt(sum(delta_sq,2)));

    % obtain the area calculation corresponding to label 'k'
    area = stats(k).Area;

    % mark objects above the threshold with a black circle
    if 4*pi*area/perimeter^2 < threshold
        % Create a list of centers and a list of radii
        if exist('centroid','var')
            centroid = [centroid;stats(k).Centroid];
            radius=[radius;stats(k).MajorAxisLength/2];
        else
            centroid = stats(k).Centroid;
            radius=stats(k).MajorAxisLength/2;
        end
    else
        %if the area is huge ignore it.
        %if the area is not huge try and determine if this is multiple beads next to each other.
        if area<2000
            %take the screen shot and break it into subimages...This is done by finding the edges
            extrem= stats(k).Extrema;
            diffy=max(round((33-max(extrem(:,2))+min(extrem(:,2)))/2),0);
            diffx=max(round((33-max(extrem(:,1))+min(extrem(:,1)))/2),0);
            starty=max(round(min(extrem(:,2)))-diffy,1);
            startx=max(round(min(extrem(:,1)))-diffx,1);
            endy=min(round(max(extrem(:,2)))+diffy,height);
            endx=min(round(max(extrem(:,1)))+diffx,width);
            %if the box made is small increase the size
you will add or subtract depending on the location of the particle relative to the edge of the image.

if endy - starty < 32
    if starty + 32 < height
        endy = starty + 32;
    else
        starty = endy - 32;
    end
end

if endx - startx < 32
    if startx + 32 < width
        endx = startx + 32;
    else
        startx = endx - 32;
    end
end

% crop image to sub area
subsection = bw(starty:endy, startx:endx);
rawimg = im2uint8(subsection);

http://www.mathworks.com/matlabcentral/fileexch
% change/9168-detect-circles-with-various-radii-in-grayscale-image-via-hough-transform/content/CircularHough_Grd.m

% finds circle centers and radii of multiple overlapping circles

% sometimes gets filter size errors try adjusting

% level, adjusting channel edges or screen boundaries
[~, circen, cirrad] = CircularHough_Grd(rawimg, [round(stats(k).MinorAxisLength/2)-1, round(stats(k).MinorAxisLength/2)+2]);

if isempty(circen)==1 || isempty(cirrad)==1
    graindata = regionprops(subsection, 'all');
    circen = cat(1, graindata.Centroid);
    cirrad = cat(1, graindata.MajorAxisLength)/2;
end

% Add updated radii to radii list
if exist('centroidmult','var')
    centroidmult = [centroidmult; circen(:,1)+startx, circen(:,2)+starty];
    radiusmult = [radiusmult; cirrad];
else
    centroidmult = [circen(:,1)+startx, circen(:,2)+starty];
    radiusmult = cirrad;
end

%plot each subsection
if nargin==6
    figure(7)
    imshow(subsection)
    set(gcf,'name','Subsection','numbertitle','off')
    hold on;
    plot(circen(:,1), circen(:,2), 'r+');
    for m = 1 : size(circen, 1),
        DrawCircle(circen(m,1), circen(m,2), cirrad(m), 32, 'b-');
    end
    hold off;
    %pause %turn on if want to view each
    press a
    %key to continue
end
end
end

%combine radii from two different techniques
if exist('centroidmult','var')
    if exist('centroid','var')
        pkn=[centroidmult;centroid];
        rads=[radiusmult;radius];
    else
        pkn=centroidmult;
        rads=radiusmult;
    end
else
    if exist('centroid','var')
        pkn=centroid;
        rads=radius;
    end
end

%If single frame plot the black and white image to add
"particle markers" to.
if nargin==6
    figure(8)
    colormap('gray'), imagesc(bw);
    set(gcf,'name','Background with particles marked','numbertitle','off')
    hold on
end
%if there are particles in the current frame that are within
the
%radii of another particle remove the second particle from
the list
%this was an attempt to get rid of "extra" particles
if exist('pkn','var')
    [m,~]=size(pkn);
    stepcen=1;
    if m>1
        while stepcen<m
            stepcen2=stepcen+1;
            while stepcen2<m+1
                if sqrt((pkn(stepcen,1)-
pkn(stepcen2,1))^2+(pkn(stepcen,2)-pkn(stepcen2,2))^2)<rads(stepcen)
                    if stepcen>1
                        rads=[rads(1:stepcen-
1,:);rads(stepcen+1:end,:)];
                        pkn=[pkn(1:stepcen-
1,:);pkn(stepcen+1:end,:)];
                    else
                        rads=rads(stepcen+1:end,:);
                        pkn=pkn(stepcen+1:end,:);
                    end
                else
                    stepcen2=stepcen2+1;
                end
            m2=m;
            [m,~]=size(pkn);
            end
        if m2==m
            stepcen=stepcen+1;
        end
    end
end
%if there are still particles after this plot there
locations
%for the single frame and save them to a list of all the
beads
%in all the frames
if m>0
    if nargin==6
        scatter(pkn(:,1),pkn(:,2))
    else
        frames=pkn(:,1)*0+frame;
        if exist('vals','var')
            vals=[vals; [pkn frames]];
            radcen=[radcen;[pkn rads]];
        else
            vals=[pkn frames];
            radcen=[pkn rads];
        end
    end
end
%finish if single frame otherwise create track
if nargin==6
    hold off
    values=[];
    isitdone=1;
else
    %if there is a list of points first check to see if there are
    %objects that don't move at all and remove those
    if exist('vals','var')
        [m,~]=size(vals);
        repeatpoints=1;
        while repeatpoints<m-1
            repeatpoints2=repeatpoints+1;
            while repeatpoints2<m
                [m,~]=size(vals);
                if vals(repeatpoints,1)== vals(repeatpoints2,1)
                    if repeatpoints>1
                        vals=[vals(1:repeatpoints2-1,:);vals(repeatpoints2+1:end,:)];
                    else
                        vals=vals(repeatpoints2+1:end,:);
                    end
                else
                    repeatpoints2=repeatpoints2+1;
                end
            end
            repeatpoints=repeatpoints+1;
        end
        %tracks is d/led m-file from matworks database.
        res=track(vals,25); %parameter #2 sets the minimum distance a particle travels between frames.
        %if the particle travels farther then value it is established
        %as a new particle. This value can be important if flow is
        %fast/slow
        [~,widthres]=size(res);
        %check to see if trajectories found
        if widthres==1
            return;
        else
            %if they are found plot a background to put the
            %trajectories on
            colormap('gray'), imagesc(B);
        %finds the locations of the first frame of all the
% trajectories within the trajectory list
Currentstart=1;
Startlist=1;
for i=1:size(res,1)
    if res(i,4)>Currentstart
        if i>1
            Startlist=[Startlist i];
            Currentstart=Currentstart+1;
        end
    end
end

% Collect data for each trajectory
for i=1:length(Startlist)
    frame=res(Startlist(i),3);
    % determine the number of frames and the number of actual points in each trajectory
    if i==length(Startlist)
        numberframes=res(size(res,1),3)-frame;
        numberpoints=size(res,1)-Startlist(i);
    else
        numberframes=res(Startlist(i+1)-1,3)-frame;
        numberpoints=Startlist(i+1)-1-Startlist(i);
    end
    % Check to see if the trajectory is likely noise
    if not
        % find important info
        if numberpoints>10 % Minimum track length - you might want to change depending on speed
            % determine the location of the end of a trajectory
            % in the list of trajectories
            if i<length(Startlist)
                stopper=Startlist(i+1)-1;
            else
                stopper=size(res,1);
            end
            % find last frame
            if exist('stops', 'var')
                stops=[stops res(stopper,3)];
            else
                stops=res(stopper,3);
            end
            % find x position at last frame
            if exist('stopx', 'var')
                stopx=[stopx res(stopper,1)];
            else
stopx=res(stopper,1);

end

%find y position at last frame
if exist('stopy','var')
  stopy=[stopy res(stopper,2)];
else
  stopy=res(stopper,2);
end

%keep track of which trajectories used
if exist('TrajUsed','var')
  TrajUsed=[TrajUsed Startlist(i)];
else
  TrajUsed=Startlist(i);
end

%find trajectory lengths
if exist('lengths','var')
  lengths=[lengths numberframes];
else
  lengths=numberframes;
end

%find start frame
if exist('starts','var')
  starts=[starts res(Startlist(i),3)];
else
  starts=res(Startlist(i),3);
end

%find x velocity
if exist('startvx','var')
  startvx=[startvx res(Startlist(i),1)];
else
  startvx=res(Startlist(i),1);
end

%find y velocity
if exist('startvy','var')
  startvy=[startvy res(Startlist(i),2)];
else
  startvy=res(Startlist(i),2);
end

%Correlate size of particle and trajectory
numfram=0;
for stepfram=Startlist(i):stopper
  xfram=res(stepfram,1);
  yfram=res(stepfram,2);
  for valstep=1:length(radcen)
    if radcen(valstep,1)==xfram
      numfram=numfram+1;
    end
  end
end
if radcen(valstep,2)==yfram
    if numfram==0
        numfram=1;
    else
        sumrad=[sumrad
                    radcen(valstep,3)];
        numfram=numfram+1;
    end
    end
end

% find average size of bead over whole trajectory
% and add to list of sizes
total=mean(sumrad);
if exist('sizes','var')
    sizes=[sizes total];
else
    sizes=total;
end

% plot each size in a different color values
% on contrast so they may need to be adjusted
hold on
if total<5
    color='r';
elsif total<=8
    color='m';
elsif total<=12
    color='g';
else
    color='b';
end

% plot trajectories
if i<length(Startlist)
    plot(res(Startlist(i):Startlist(i+1)-1,1),res(Startlist(i):Startlist(i+1)-1,2),
         'color',color)
else
    plot(res(Startlist(i):size(res,1),1),res(Startlist(i):size(res,1),2),
         'color',color)
end
%determine if particle sorted
if res(stopper,1)>rightx
    sorted=1;
    if res(Startlist(i),1)>rightx
        sorted=1.25;
    end
else
    if res(stopper,2)<topy
        sorted=0.5;
    else
        sorted=0;
        if res(Startlist(i),1)<leftx
            sorted=0.25;
        end
        if res(stopper,1)>leftx
            sorted=0.75;
        end
    end
end
end

%record if it sorted
if exist('sorts','var')
    sorts=[sorts sorted];
else
    sorts=sorted;
end

muperpix=0.235;%micrometers per pixel
frpersec=19.57;%frames per second

%find velocity before intersection
for k=Startlist(i):stopper
    if res(k,1)>rightx
        vbefore=0;
    else
        valatrightx=k;
        vbefore=muperpix*frpersec*(res(valatrightx,1)-res(Startlist(i),1))/(res(Startlist(i),3)-res(valatrightx,3));
        break;
    end
end

%find velocity in intersection
for k=Startlist(i):stopper
    if res(k,1)<rightx
        vduring=0;
    else
        valatstartafter=k;
        vduring=muperpix*frpersec*(res(valatstartafter,1)-res(stopper,1))/(res(stopper,3)-res(valatstartafter,3));
    end
end
break;
end
end

% find velocity after intersection
if sorted == 1
  for k=Startlist(i):stopper
    if res(k,2) > topy
      vafter=0;
    else
      valatstartafter=k;
    vafter=muperpix*frpersec*(res(valatstartafter,2) - res(stopper,2))/(res(stopper,3) - res(valatstartafter,3));
    break;
  end
else
  for k=Startlist(i):stopper
    if res(k,1) > leftx
      vafter=0;
    else
      valatstartafter=k;
    vafter=muperpix*frpersec*(res(valatstartafter,1) - res(stopper,1))/(res(stopper,3) - res(valatstartafter,3));
    break;
  end
end

% record velocity before
if exist('vbefores','var')
  vbeforees=[vbeforees vbefore];
else
  vbeforees=vbefore;
end
% record velocity during
if exist('vdurings','var')
  vbeforees=[vdurings vduring];
else
  vduring=vduring;
end
% record velocity after
if exist('vafters','var')
  vafters=[vafters vafter];
else
  vafters=vafter;
end
% record filenum so can combine info later
if exist('filenumlist','var')  
    filenumlist=[filenumlist filenum];  
else  
    filenumlist=filenum;  
end

if exist('sizeestimate','var')  
    sizeestimate=[sizeestimate 0]; %actual estimate added later just a place holder  
else  
    sizeestimate=0;  
end

%combine information into one file
values=[starts; stops; sizes; sorts; vbefores; vduring; vafters; sizeestimate; startvx; startvy; stopx; stopy; filenumlist];

end

hold off

%this is used to find statistics for each size
before and after sorting region
if exist('values','var')
    point5velobefor=0;  
    point5veloafatersort=0;  
    point5veloafter=0;  
    point5sort=0;  
    point5vals=0;  
    com1velobefor=0;  
    com1veloafatersort=0;  
    com1veloafter=0;  
    com1sort=0;  
    com1vals=0;  
    com2velobefor=0;  
    com2veloafatersort=0;  
    com2veloafter=0;  
    com2sort=0;  
    com2vals=0;  
    com3velobefor=0;  
    com3veloafatersort=0;  
    com3veloafter=0;  
    com3sort=0;  
    com3vals=0;  
    [width,height]=size(values);  
    for i=1:height  
        if(isnan(values(5,i))==1)  
            values(5,i)=0;  
        end
end
if values(3,i)< 6.5
    values(7,i)=0.5;
    point5vals=point5vals+1;
    point5velobefor=values(5,i)+point5velobefor;
    if(values(4,i)==1)
        point5veloaftersort=values(6,i)+point5veloaftersort;
        point5sort=point5sort+1;
    else
        point5veloafter=values(6,i)+point5veloafter;
    end
else
    if values(3,i)<=8
        values(7,i)=1;
        com1vals=com1vals+1;
        com1velobefor=values(5,i)+com1velobefor;
        if(values(4,i)==1)
            com1veloaftersort=values(6,i)+com1veloaftersort;
            com1sort=com1sort+1;
        else
            com1veloafter=values(6,i)+com1veloafter;
        end
    else
        if values(3,i)<=12
            values(7,i)=2;
            com2vals=com2vals+1;
            com2velobefor=values(5,i)+com2velobefor;
            if(values(4,i)==1)
                com2veloaftersort=values(6,i)+com2veloaftersort;
                com2sort=com2sort+1;
            else
                com2veloafter=values(6,i)+com2veloafter;
            end
        else
            if values(3,i)>12
                values(7,i)=3;
                com3vals=com3vals+1;
                com3velobefor=values(5,i)+com3velobefor;
                if(values(4,i)==1)
                    com3veloaftersort=values(6,i)+com3veloaftersort;
                    com3sort=com3sort+1;
                else
                    com3veloafter=values(6,i)+com3veloafter;
                end
        end
end
com3veloafter=values(6,i)+com3veloafter;
        end
        end
    end
end

NumberOfBeads=[point5vals,com1vals,com2vals,com3vals];

Velocitybeforesorting=[point5velobefor,com1velobefor,com2velobefor,com3velobefor]./NumberOfBeads;

Velocityaftersortingifnotsorted=[point5veloafter,com1veloafter,com2veloafter,com3veloafter]./NumberOfBeads;

PercentSort=[point5sort,com1sort,com2sort,com3sort]./NumberOfBeads*100;

Velocityaftersortingifsorted=[point5veloaftersort,com1veloaftersort,com2veloaftersort,com3veloaftersort]./NumberOfBeads;

stats=[0.5,1,2,3;NumberOfBeads;PercentSort;Velocitybeforesorting;Velocityaftersortingifnotsorted;Velocityaftersortingifsorted];

% save everything!
if exist('allvals2.mat', 'file')==2
    load allvals2
end
if exist('allvalues','var')==0
    allvalues=[];
end
startfilename=max(strfind(filename,'/'));
if isempty(startfilename);
    startfilename=0;
end
filename2=filename(startfilename+1:end);
filename2=strrep(filename2,'.','');
matname=sprintf('Data%s.mat',filename2);
save(matname)
figurename=sprintf('Trajectory%s',filename2);
saveas(gcf,figurename,'fig')
saveas(gcf,figurename,'jpg')

end
isitdone=1; % done!
end
end
Tracking software for low movement

function
ParticleTrackFunctionTrap(filename, rotate, crop, Guide, ParticleLocation, splitvalue, frame, frameend)
%Plots trajectories of particles in waveguide.
%Code by Kaelyn Leake
% for a single frame:
ParticleTrackFunctionTrap(filename, rotate, crop, Guide, ParticleLocation, splitvalue, frame)
% for all frames:
ParticleTrackFunction(filename, rotate, crop, Guide, ParticleLocation, splitvalue)
% for range of frames:
ParticleTrackFunction(filename, rotate, crop, Guide, ParticleLocation, splitvalue, frame, frameend)
% filename should be in form of string it excepts tifs ex.
'C:/Desktop/130728'
% rotate is 1 if you want to rotate the video else 0 --> orintation
% expected __|__ or
% crop is 0 if you don't want to crop otherwise it's
[xmincrop, ymincrop, widthcrop, heightcrop]
% Guide is [topy, bottomy, leftx, rightx] an ex: [511, 567, 206, 260] for
% intersection....[topy, bottomy] an ex: [511, 567] for straight
% guide...0 if
% whole image
% ParticleLocation is [cropx, cropdx, cropy, cropdy] or 0 if you want
to
% select on figure
% splitvalue is 0 if you want to chose a point or the intensity
value of
% the color in which >white and <black
clc
close all

%initialize constants
boltz=1.3806503e-23;
T=293;
valsrad=[];
muperpix=0.175644;
stiffness=[];
valso=[];
potential=[];

gfigsavelcation = strrep(filename,'_','');
gfigsavelcation = strrep(figsavelcation,'.', '');
gfigsavelcation = [figsavelcation, '/ '];

% Define Waveguide
if length(Guide) == 4 % if intersection
topy = Guide(1);
bottomy = Guide(2);
leftx = Guide(3);
rightx = Guide(4);
elseif length(Guide) == 2 % if straight channel

topy = Guide(1);
bottomy = Guide(2);
end

% Crop and then define the waveguide
if length(crop) == 4
    xmincrop = crop(1);
ymincrop = crop(2);
widthcrop = crop(3);
heightcrop = crop(4);
    if length(Guide) == 4
        topy = topy - ymincrop;
        bottomy = bottomy - ymincrop;
        leftx = leftx - xmincrop;
        rightx = rightx - xmincrop;
        elseif length(Guide) == 2
            topy = topy - ymincrop;
            bottomy = bottomy - ymincrop;
            leftx = xmincrop;
            rightx = xmincrop + widthcrop;
    end
end

% display file info on the m-file
display('The Program is Now Running!')
fslash = strfind(filename, '/ ');
filenum = str2double(filename(fslash(end) + 1:end));
fname = sprintf('%s.tif', filename);

% Get File info and open file
info = imfinfo(fname);
B = imread(fname, frame, 'Info', info);

% check to see if any rotation is necessary
if rotate == 1
    B = flipud(rot90(B));
    B = flipud(B);
end
%do cropping if required
if length(crop)==4
    B=imcrop(B,[xmincrop,ymincrop,widthcrop,heightcrop]);
end

[height,width]=size(B);

%mark bead area
if length(ParticleLocation)~=4
    figure(1)
    colormap('gray'), imagesc(B);
    set(gcf,'name','Choose Particle Location!','numbertitle','off')
    [X,Y]=ginput(2);
    cropx=min([X(1),X(2)]);
    cropdx=abs(X(1)-X(2));
    cropy=min([Y(1),Y(2)]);
    cropdy=abs(Y(1)-Y(2));
    set(gcf,'name','Original Frame','numbertitle','off')
else
    figure(1)
    colormap('gray'), imagesc(B);
    set(gcf,'name','Original Frame','numbertitle','off')
    cropx=ParticleLocation(1);
    cropdx=ParticleLocation(2);
    cropy=ParticleLocation(3);
    cropdy=ParticleLocation(4);
end

%create boundaries for none given values
if length(crop)~=4
    if length(Guide)~=4
        leftx=1;
        rightx=width;
        if length(Guide)~=2
            topy=height;
            bottomy=1;
        end
    end
end

%determine range of file
if nargin==8
    startframe=frame;
    stopframe=frameend;
    framerange=startframe:stopframe;
elseif nargin==6
    startframe=1;
end
stopframe=numel(info);
  framerange=startframe:stopframe;
else
  framerange=frame;
end

if nargin~=7
  newimage=zeros(height,width,stopframe); %to much for memory some
  computers may need to comment this and line 215
end

for frame=framerange
  %This function takes an image and adjusts contrasts and
  identifies objects
  A= imread(fname, frame, 'Info', info);

  if rotate==1 %rotate
    A=flipud(rot90(A));
    A=flipud(A);
  end
  maxA=max(max(A));

  %white out non-channel area-whites out areas of image to speed
  up run
  %time -> will become black after splitvalue
  if length(Guide)==4
    A(1:topy,1:leftx)=maxA;
    A(1:topy,rightx:width)=maxA;
    A(bottomy:height,1:width)=maxA;
  elseif length(Guide)==2
    A(1:topy,1:width)=maxA;
    A(bottomy:height,1:width)=maxA;
  end

  %crop to want area
  if length(crop)==4 %crop
    A=imcrop(A,[xmincrop,ymincrop,widthcrop,heightcrop]);
  end

  %Select region of interest
  B=imcrop(A,[cropx,cropy,cropdx,cropdy]);

  %contrast adjustment values,larger than splitvalue become white
  and
  %smaller values become black
  if splitvalue==0
    figure(3)
    colormap('gray'), imagesc(B);
    set(gcf,'name','Choose splitvalue!','numbertitle','off')
    [X,Y]=ginput(1);
    splitvalue=B(round(Y),round(X));
    close(3)
end
C=B;
C(C>splitvalue)=65530;
C(C<=splitvalue)=100;

%comment these if want light objects rather than dark ones
C(C==65530)=0;
C(C==100)=65530;

level = 0.5;
bwa = im2bw(C); %convert ot black and white
bwb = imfill(bwa,'holes');%fill holes in particles
bw = bwareaopen(bwb, 200); %get rid of small things. May need to
adjust depending on contrast and size of beads

%plot if single frame
if nargin==7
figure(2)
colormap('gray'), imagesc(A);
set(gcf,'name','With waveguides
removed','numbertitle','off')
figure(3)
colormap('gray'), imagesc(B);
set(gcf,'name','Region of interest','numbertitle','off')
figure(4)
colormap('gray'), imagesc(C);
set(gcf,'name','Region of interest after values
split','numbertitle','off')
figure(5)
colormap('gray'), imagesc(bwa);
set(gcf,'name','Black and white','numbertitle','off')
figure(6)
colormap('gray'), imagesc(bw);
set(gcf,'name','Black and white remove
small','numbertitle','off')
else
 newimage(:,:,frame-startframe+1)=bw;
end

%Find objects and get statistics
[Bs,L] = bwboundaries(bw,'noholes');
stats =
 regionprops(L,'Area','Centroid','MinorAxisLength','Extrema','MajorAxisLength');
for k = 1:length(Bs)
  if exist('valscen')
    valscen = [valscen;stats(k).Centroid,frame];
    valsrad=[valsrad;stats(k).MajorAxisLength/2];
  else
    valscen = [stats(k).Centroid,frame];
    valsrad=stats(k).MajorAxisLength/2;
  end
end
end
if nargin==7
valsn=[valscen(:,1)+cropx,valscen(:,2)+cropy,valscen(:,3)]; % put
back into full image
valscen=valsn;
valsum=[valscen(:,1:2)*0.175644,valscen(:,3)]; % put into um

% Tracking parameters
param.good=10;
param.dim=(stopframe-startframe)/10;
param.quiet=0;
res=track(valscen,20,param);

% do for all trajectories
for i=1:max(res(:,4))
% find info for current trajectory
trac=res(res(:,4)==i,:);
% save its trajectory
eval(['valsn' num2str(i) '= trac;']);
eval(['valsumn' num2str(i) '(:,1:2)*0.1739,valsumn' num2str(i) '(:,3)];']);
% find its stiffness and add to list of stiffnesses
eval(['stiffnesst=[10^12*boltz*T./var(valsumn' num2str(i) '(:,1))];10^12*boltz*T./var(valsumn' num2str(i) '(:,2));']);
% at one point had this as N/m only on row one so if power is wrong
stiffness=[stiffness,stiffnesst];

% find potential of this trajectory
eval(['potentialt=(xm' num2str(i) '^2*stiffness(1,1)+ym' num2str(i) '^2*stiffness(2,1))./(2*boltz*293).*10^-12;']);
potential=[potential,potentialt];
% recenter the data about center of trajectory
eval(['valsumo' num2str(i) '=[valsumn' num2str(i) '(:,1)-median(valsumn' num2str(i) '(:,1)),valsumn' num2str(i) '(:,2)-median(valsumn' num2str(i) '(:,2)),valsumn' num2str(i) '(:,3)];']);
eval(['res' num2str(i) '=trac;']);
eval(['valso=[valso;valsumo' num2str(i) '];']);

% plot bead's x histogram
figure
eval(['set(gcf,''name'', ''bead ' num2str(i) ' - x - histogram'', ''numbertitle'', ''off'');']);
% plot bead's y histogram
figure
eval(['set(gcf,''name'', ''bead ' num2str(i) ' - y - histogram'';'',''numbertitle'';'',''off'');']);
xlabel('Y Position (\mum)');
ylabel('Counts');
eval(['title(''Bead ' num2str(i) ''');']);
eval(['saveas(gcf,''figsavelocation 'bead ' num2str(i) ' - y - histogram.fig'');']);

% plot each bead's potential
figure
eval(['set(gcf,''name'', ''bead ' num2str(i) ' - xy - potential'';'',''numbertitle'';'',''off'');']);
xb = linspace(min(valsumo' num2str(i) '(:,1)),max(valsumo' num2str(i) '(:,1)),5);
yb = linspace(min(valsumo' num2str(i) '(:,2)),max(valsumo' num2str(i) '(:,2)),5);
contourf(xb,yb,(n'-max(max(n)))/max(max(n))*-1*potential(1),7)
eval(['title(''Bead ' num2str(i) ''');']);
xlabel('X Position Offset (\mum)');
ylabel('Y Position Offset (\mum)');
c=colorbar;
ylabel(c,'Potential (kT)');
eval(['saveas(gcf,''figsavelocation 'bead ' num2str(i) ' - xy - potential.fig'');']);
end

% The following plots are for all the beads or the full area can be
% fully commented out or individually or as needed
figure
set(gcf,'name','All beads - xy - Histogram
Combined','numbertitle','off')
xb = linspace(min(valso(:,1)),max(valso(:,1)),5);
yb = linspace(min(valso(:,2)),max(valso(:,2)),5);
n=double(hist3(valso(:,1:2),[5,5]));
contourf(xb,yb,n',9)
title('Combined Offset Histogram ')
c=colorbar;
ylabel(c,'Counts')
saveas(gcf, [figsavelcation,'All beads - xy - Histogram Combined.fig']);

figure
set(gcf,'name','Full area - xy - Histogram','numbertitle','off')
xb = linspace(min(valsnum(:,1)),max(valsnum(:,1)),120);
yb = linspace(min(valsnum(:,2)),max(valsnum(:,2)),5);
n=double(hist3(valsnum(:,1:2),[120,5]));
contourf(xb,yb,n',9)
title('Histogram')
c=colorbar;
ylabel(c,'Counts')
saveas(gcf, [figsavelcation,'Full area - xy - Histogram.fig']);

potentialmat=(n'-max(max(n)))/max(max(n))*-1*max(potential);
dxf=diff(potentialmat,1,1);
dyf=diff(potentialmat,1,2);
mask = ones(1,2);
adxf = conv2(dxf,mask,'valid')/2;
axb=conv2(xb,mask,'valid')/2;
ayb=conv2(yb,mask,'valid')/2;
mask = ones(2,1);
adyf = conv2(dyf,mask,'valid')/2;
fx=-1.*adxf./mean(diff(xb));
fy=-1.*adyf./mean(diff(yb));
F=sqrt((fy).^2+(fx).^2);

figure
set(gcf,'name','Full area - Total Force','numbertitle','off')
contourf(axb,ayb,F,9)
title('Total Force');
xlabel('X Position Offset(\mum)');
ylabel('Y Position Offset(\mum)');
c=colorbar;
ylabel(c,'Total Force(kT/\mum)')
saveas(gcf, [figsavelcation,'Full area - Total Force.fig']);

figure
set(gcf,'name','Full area - X Force','numbertitle','off')
contourf(axb,ayb,fx,9)
title('X Force');
xlabel('X Position Offset(\mum)');
ylabel('Y Position Offset(\mum)');
c=colorbar;
ylabel(c,'X Force(kT/\mum)')
saveas(gcf, [figsavelcation,'Full area - X Force.fig']);

figure
set(gcf,'name','Full area - Y Force','numbertitle','off')
contourf(axb,ayb,fy,9)
title('Y Force');
xlabel('X Position Offset(\mum)');
ylabel('Y Position Offset (\mum)');
c=colorbar;
ylabel(c,'Y Force (kT/\mum)')
saveas(gcf, [figsavelocation,'Full area - Y Force.fig']);

figure
set(gcf,'name','Full area - Scatter','numbertitle','off')
A= imread(fname, startframe, 'Info', info);
imshow(A)
hold on
for i=1:max(res(:,4))
    eval(['scatter(res' num2str(i) '(:,1),res' num2str(i) '(:,2));']);
end
hold off
saveas(gcf, [figsavelocation,'Full area - Scatter.fig']);

figure
set(gcf,'name','All beads - y - Histogram','numbertitle','off')
hold on
for i=1:max(res(:,4))
    eval(['hist(valsumn' num2str(i) '(:,2));']);
end
hold off
xlabel('Y Position (\mum)');
ylabel('Counts');
saveas(gcf, [figsavelocation,'All beads - y - Histogram.fig']);

figure
set(gcf,'name','All beads - x offset - Histogram','numbertitle','off')
hold on
for i=1:max(res(:,4))
    eval(['hist(valsumo' num2str(i) '(:,1));']);
end
hold off
xlabel('X Position Offset (\mum)');
ylabel('Counts');
saveas(gcf, [figsavelocation,'All beads - x offset - Histogram.fig']);

figure
set(gcf,'name','All beads - y offset - Histogram','numbertitle','off')
hold on
for i=1:max(res(:,4))
    eval(['hist(valsumo' num2str(i) '(:,2));']);
end
hold off
xlabel('Y Position Offset (\mum)');
ylabel('Counts');
saveas(gcf, [figsavelocation,'All beads - y offset - Histogram.fig']);
figure
set(gca,'name','Full area - Trajectory','numbertitle','off')
A= imread(fname, startframe, 'Info', info);
imshow(A)
hold on
for i=1:max(res(:,4))
    eval(['plot(res num2str(i) '(:,1),res num2str(i) '(:,2));']);
end
hold off
title('Trajectory')
saveas(gcf, [figsavelcation,'Full area - Trajectory.fig']);

figure
set(gca,'name','Full area -xy- Potential','numbertitle','off')
xb = linspace(min(valsnum(:,1)),max(valsnum(:,1)),140);
yb = linspace(min(valsnum(:,2)),max(valsnum(:,2)),10);
n=hist3(valsnum(:,1:2),[140,10]);
contourf(xb,yb,(n'-max(max(n)))/max(max(n))*-1*max(potential),9)
xlabel('X Position Offset(\mum)');
ylabel('Y Position Offset(\mum)');
c=colorbar;
ylabel(c,'Potential(kT)')
saveas(gca, [figsavelcation,'Full area -xy- Potential.fig']);
end

save([figsavelcation,'data.mat']); %save the data
11 Bibliography


