

©Copyright 2025

Tommy Zhang

Automated Transport of Micro-Objects With Complex Trap Configurations in Holographic Optical Tweezers

Tommy Zhang

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

Master of Science

University of Washington

2025

Committee:

Ashis G. Banerjee

Santosh Devasia

Sawyer Fuller

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Automated Transport of Micro-Objects With Complex Trap Configurations in
Holographic Optical Tweezers

Tommy Zhang

Chair of the Supervisory Committee:
Ashis G. Banerjee
Mechanical Engineering

Optical tweezers use a focused laser beam to trap micro and nano-scale objects through a scattering and gradient force exhibited due to the properties of light. A specialized holographic optical tweezer (HOT) which is centered around the use of a spatial light modulator can generate multiple optical traps in the workspace through holograms. HOTs have found many use cases in micro-robotics and the medical field, where they have been used for cell characterization, cell-to-cell interactions, and micro-robotic actuators.

We will begin by presenting a HOT specific framework on which to build control systems intended to augment user control of HOTs. We then integrate the framework with our HOT system and develop a model predictive control (MPC) based adaptive path planning control system capable of transporting polystyrene micro-beads to create formations in the workspace. We develop the MPC based system to consider multiple trap configurations including point, annular, and line traps. The latter two of which can be used to trap multiple micro-beads in a ring or line configuration respectively. The control system leverages GPU compute and Python parallelization to allow for simultaneous path planning and control for multiple optical traps. We then demonstrate the effectiveness of the controller by allowing it to create various micro-object formations automatically while avoiding collisions with obstacles in a crowded workspace.

Then, we will focus on the phase retrieval process, another area of research in the HOT

field and other applications involving the use of holograms. Phase retrieval is the process of calculating a phase mask for addressing onto an SLM to form a desired hologram, or in the case of HOTs, an intensity distribution in the image plane. We investigate the capability for deep learning models, specifically the convolutional neural network architecture U-Net, to act as the mechanism for phase calculation during operation of the HOT system. By using Red Tweezers, an open-source hologram engine built for calculating phase masks on our HOT system, for creating synthetic training data. Then by carefully choosing hyperparameters and training data distributions we train the model on 2D intensity distributions of the far-field and the phase masks generated by Red Tweezers. We then conducted trials on our HOT system and demonstrate the capability of the U-Net to act as the phase retriever in a single pass. The U-Net allows for constant phase calculation times, whereas the Red Tweezers algorithm would be slowed to update rates insufficient for real-time control using large numbers of optical traps.

We aim for the methodologies developed for both of these topics to be useful for further research into autonomy integrated HOT systems.

TABLE OF CONTENTS

	Page
List of Figures	iii
Chapter 1: Introduction	1
1.1 Principle of Optical Tweezers	1
1.2 Holographic Optical Tweezers	1
1.3 Control in Stochastic Environments	4
Chapter 2: Automated Transport of Composite Micro-Objects	5
2.1 Methods	5
2.1.1 HOT System	5
Optical Traps	6
2.1.2 Problem Formulation	7
2.1.3 Iterative Linear Quadratic Regulator	8
iLQR Formulation	9
iLQR Solve	10
2.1.4 Model Predictive Control	12
2.1.5 Obstacle Model	12
Minimum Distance from a Point to the Edge of an Ellipse	14
DBSCAN Clustering	15
2.1.6 Graphical Interface	17
2.1.7 Integration of Multiple Controlled Beads	18
2.2 Results	18
2.2.1 Implementation	18
Trap Intensity Scaling	18
JAX Parallelization	20
MPC Parameters	21
2.2.2 Experimental Setup	22
2.2.3 Experiment Execution	22
2.2.4 Experiments	23

2.2.5	Experimental Results	23
2.3	Discussion	25
Chapter 3:	Deep Learning Based Hologram Generation	30
3.1	Methods	31
3.1.1	Model Architecture	31
3.1.2	Training Data	32
3.1.3	Technical Approach	34
3.1.4	Execution	36
3.2	Experimental Evaluation	36
3.2.1	Experimental Setup	36
3.2.2	Results	38
3.3	Discussion	39
3.4	Conclusions	44
Chapter 4:	Conclusions	45
4.1	Review of Contributions	45
4.2	Anticipated Impact	45
4.3	Future Work	46
Bibliography		48

LIST OF FIGURES

Figure Number		Page
1.1	Illustration of the gradient force and radiation pressure (scattering force) experienced by a dielectric micro-sphere in an optical tweezers system. The laser beam is focused using optics to a diffraction limited point in red. The blue circle indicates the micro-sphere experiencing a gradient force towards the center of the beam and a scattering force axially in the direction of the laser propagation.	2
2.1	Screenshot of phase patterns and trap back reflection for the various trap types. a), b), c) are the line, annular, and point trap phase patterns respectively. d), e), f) are the imaged back reflections of the line, annular, and point traps when a mirror is placed in the sample plane.	6
2.2	Problem formulation illustration. Red circles with jagged arrows represent freely diffusing obstacles exhibiting Brownian motion. Other shaded circles represent trapped beads. The objective is to simultaneously maneuver all trapped beads to their respective shaded goal positions without collisions. . .	9
2.3	Pre-processing steps for ellipse distance algorithm. a) original state of obstacles relative to the line trap. b) coordinate system origin translated to the center point of line trap. c) absolute value of obstacle coordinates taken to ensure all stay in first quadrant. d) relative obstacle positions after pre-processing.	16
2.4	Screenshot of the optical tweezers user interface. Line traps are denoted by a purple line, annular traps with a large blue circle, and point traps with a green circle. The goal positions for the line, annular, and point trap are the small green circle, orange concentric circles, and green circle respectively. . .	17
2.5	Overview of the HOT automation framework. The main allocator is run on startup and maintains the state of the entire workspace. Once the operator has traps and goals placed, sliders are toggled to start a separate control process for each trap-goal combination.	19
2.6	Depiction of the various classes of bead formations assembled for experiments. a) is a formation comprised of both line and annular traps in a dumbbell-like pattern. b) consists of point and line traps assembled to represent a diamond. c) contains all three types of traps and forms a parabolic arc.	23

2.7	Experiment with annular, line, and point traps. Starting and goal positions for each trap type are given at $t=0s$. $t=98s$ shows the end of the experiment along with the overall trajectory for each trap.	24
2.8	Experiment with line and point traps. Starting and goal positions for each trap type are given at $t=0s$. $t=58s$ shows the end of the experiment along with the overall trajectory for each trap.	25
2.9	Experiment with annular and line traps. Starting and goal positions for each trap type are given at $t=0s$. $t=56s$ shows the end of the experiment along with the overall trajectory for each trap.	26
2.10	Stair step pattern formation. The pattern was assembled in two stages, the first being all three line traps and second being the two point traps.	29
3.1	U-Net architecture for phase mask generation in holographic optical tweezers	31
3.2	Flowchart highlighting the difference between the U-Net and Red Tweezers phase mask calculation approaches. a) Input intensity distribution to U-Net model indicating placement of two point traps. b) Phase mask calculated using Red Tweezers. c) Phase mask prediction from U-Net model given a) as input. d) Simulated far field of b), note the presence of additional orders of diffraction. e) Simulated far field of c) indicating the correct locations of the two point traps. Red arrows indicate the corresponding point traps shown in a)	34
3.3	Phase mask calculation time comparison between Red Tweezers, Gerchberg-Saxton (GS), and U-Net methods.	37
3.4	Comparisons for a single annular trap. a) Gerchberg-Saxton (GS) generated phase mask b) Far-field simulation of a). c) Red tweezers generated phase mask. d) Far-field simulation of c). e) U-Net predicted phase mask (128×128 model). f) Far-field simulation of e)	40
3.5	Phase mask predictions for a moving line trap. a) U-Net predicted phase mask. b) Far-field simulation of a) with the red arrow indicating the correctly reproduced location of the line trap. c) and d), e) and f) are pairs of U-Net predictions and simulated far-fields of the single line trap in different locations. The red arrows indicate the region of interest containing the reproduced line trap.	41
3.6	User controlled line trap trajectory time lapse. U-Net predictions used as the phase masks displayed onto the SLM highlighted outlined in orange. Red circles highlight the trapped beads. The overall trajectory is displayed at $t = 25s$ in red.	42

- 3.7 Phase mask predictions for complex intensity profiles using $D = 128$ model.
- a) Arbitrarily drawn intensity distribution. c) U-Net output given a) as input.
 - e) Far-field simulation of c). b) Extension of a) where another spline is added arbitrarily. d) U-Net output given b) as input. f) Far-field simulation of d) . 43

ACKNOWLEDGMENTS

I would especially like to thank Professor Ashis Banerjee for his invaluable mentorship throughout this journey. His guidance during our many meetings helped me navigate the numerous challenges and setbacks involved in developing the system as it stands today. I am deeply grateful for the opportunity that he gave me to take ownership of the design and engineering of the optical tweezers automation system. The knowledge and experience gained through the numerous experiments and demonstrations have been instrumental to my growth as an engineer.

I would like to thank Professor Karen Leung for teaching inspiring courses and for giving me the opportunity to work on projects that ultimately became integrated into my research. Her teaching sparked ideas that shaped both my technical interests and my research direction.

I am also deeply appreciative of Professor John Choe for his early encouragement and support, which played a pivotal role in shaping my academic journey. His mentorship and motivation led me to pursue my interests in robotics and ultimately brought me to where I am today.

Lastly, I want to thank my mother, Jan, and my sister, Lucille, for their unwavering support throughout my entire academic journey.

DEDICATION

to my cat, Rue

Chapter 1

INTRODUCTION

1.1 Principle of Optical Tweezers

Optical tweezers are instruments capable of manipulating micro and nano-scale objects using a laser. These objects are often suspended in a fluid medium where the position of the object can be precisely controlled in all three dimensions. The main principle behind the ability of optical tweezers to manipulate these micro-objects is the scattering force of light and a gradient restoring force due to the beam geometry.

The basic principle of optical tweezers has been extended and successfully implemented in commercialized systems available for use. Many of these systems realize tweezing of multiple micro-objects concurrently, enabling a higher degree of flexibility when conducting experiments. Galvanometer based setups use mirrors in combination with a galvanometer actuator to quickly scan over the work area so a laser beam can be aimed at multiple micro-objects in a time-share arrangement [37]. These types of optical tweezers can trap multiple particles concurrently, though with some limitations due to the laser power being shared over time by multiple objects.

1.2 Holographic Optical Tweezers

Holographic optical tweezers (HOTs) are another strategy for realizing concurrent manipulation of micro-objects. HOTs leverage a spatial light modulator (SLM) which serves as a programmable phase array on which a beam of a laser is incident. As the beam reflects off the SLM, individually addressed pixels on the SLM are configured such that a controlled phase-shift is achieved as light is incident on a specific pixel. The SLM can achieve a phase-shift from $-\pi$ to π through the use of liquid crystals on a silicon backplate. By applying a varying voltage to each pixel, the crystals can be arranged at varying angles to realize the phase-shift on the reflected wave. One advantage of the SLM-based setup for optical

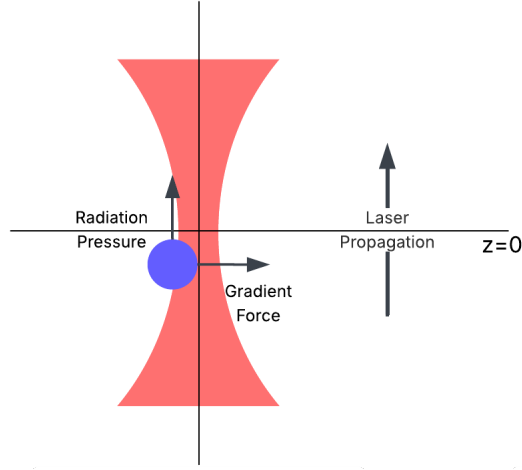


Figure 1.1: Illustration of the gradient force and radiation pressure (scattering force) experienced by a dielectric micro-sphere in an optical tweezers system. The laser beam is focused using optics to a diffraction limited point in red. The blue circle indicates the micro-sphere experiencing a gradient force towards the center of the beam and a scattering force axially in the direction of the laser propagation.

tweezers is the ability to trap large numbers of micro-objects concurrently with continuous laser power, having no mechanical systems aiming a laser beam at different points in the workspace (like the galvanometer setup), a HOT system is static in nature with no moving mechanical parts required to add, remove, or change positions of optical traps. Positioning new traps or changing the position of existing traps is analogous to updating an image on a typical LCD.

In more basic tweezing arrangements, the hologram for a single point trap is merely a diffraction-limited point of light onto which a micro-object is pulled into. However, the use of holography in tweezers setups has extended the types of objects optical tweezers can manipulate, non-dielectric particles cannot be manipulated using standard point traps, but rather require a laguerre-gaussian (LG) mode beam which is characterized by an annular-shaped intensity distribution which carries orbital angular momentum [10]. These LG beams can also be used to tweeze multiple micro-objects with a single trap and are a form of multiplexed traps.

HOT systems do come with a significant consideration when computing the phase-masks

to display on the SLM. These phase-masks are $D \times D$ pixel arrays that are displayed on the SLM to obtain a specific trapping arrangement in the workspace. Computing these phase-masks have traditionally been allocated to an iterative phase retrieval method named the Gerchberg-Saxton (GS) algorithm [14]. The GS algorithm finds the phase mask required to achieve a specified far-field diffraction pattern from a reference image plane by iteratively taking the fast fourier transform (FFT) and inverse FFT until the error in the far-field plane compared to the reference intensity is below a certain threshold. The GS algorithm, along with various extensions to 3-D are the core algorithms for computing phase masks for computational holography [8]. Although in the HOT field where fast phase mask computations are desired for real-time control of the tweezers via automation or a user interface, the GS algorithm becomes undesirable due to lengthy convergence times for high dimensional phase masks. With varying convergence times for different input intensities and no guarantee of a useful phase mask within a certain number of iterations, the GS algorithm has largely been phased out of HOT system in preference for faster, use-case specific algorithms such as the lenses and gratings algorithm used in Red Tweezers, an open source hologram calculation software for addressing SLMs in HOT applications [21, 6].

Red Tweezers implements the gratings and lenses algorithm and adds additional functionality such as generating line traps and annular traps. Phase masks displayed on the SLM are used to 'steer' and aim the incident laser beam towards the desired trap location. Multiple traps are realized by combining multiple gratings' contributions to the phase mask. The Red Tweezers hologram generation code runs locally on a GPU and takes advantage of the parallelized shader code to rapidly compute new phase masks in real-time. Real-time control strategies are possible on HOTs when using sufficiently fast phase mask calculation strategies, Red Tweezers is capable of controlling a large number of traps simultaneously without significant detriment to the computation time. However, the computation time increases proportionally to the number of traps as the phase mask is calculated by looping each SLM pixel over all the desired traps.

With the proliferation of deep learning applied to controlling complex physical systems, i.e. autonomous vehicles, humanoid robots, and spacecraft, the motivation for greater generality and flexibility in all applied controls is stronger than ever.

1.3 Control in Stochastic Environments

Stochasticity adds additional complexity and considerations to developing strategies for effective control of systems. Optical tweezers are a prime example of a stochastic system due to the brownian motion exhibited by micro-sized object in a fluid suspension. Depending on the size of the micro-object ($0.5 \mu m$ to $10 \mu m$), the motion can be more or less pronounced, where tasks requiring precise maneuvering or placement of micro-objects will experience workspace changes continuously (random displacements of untrapped micro-objects).

Modern strategies which deal with control in stochastic environments is a widely researched and broad subfield of control theory. We will only give a brief overview of more recent techniques and refer the reader to the plethora of resources available for deeper coverage of stochastic control. Many of the control strategies developed dealing with stochasticity in state observations or measurements will be under the assumption of Gaussian distributed noise for closed form analytical solutions.

In the HOT system, trapped micro-objects will be displaced to the center of the diffraction limited point trap and thus the accuracy and repeatability are purely dependent on the calibration of the optics system. Lookup tables and PID position control can be utilized to improve the absolute placement accuracy of the optically trapped object. However, in many cases only the relative positioning of multiple particles are of importance when designing experiments involving optical tweezers. Therefore, we take a deeper focus on control strategies which deal with the brownian motion of the fluid suspended micro-objects. The general system design would then follow by modeling the dynamics of the control (trap position) and the micro-object being controlled, then designing a methodology to account for continuous random displacements of the free micro-objects which we consider to be obstacles.

A popular control strategy which can deal with continuously changing constraints is model predictive control (MPC) which calculates the optimal controls for a specified time horizon, executes the first control, then repeats the calculation. In this way MPC can account for new and dynamic constraints such as moving obstacles. MPC can be realized by any optimal controller that can re-plan a new control sequence fast enough for the environment.

Chapter 2

AUTOMATED TRANSPORT OF COMPOSITE MICRO-OBJECTS

The following chapter describes an MPC control strategy used for guiding micro-particles in a HOT system to form various composite micro-structure patterns. We will also go into detail on the design of the control system for obstacle avoidance, concurrently controlled optical traps, and how we deal with the different possible optical trap types.

2.1 *Methods*

2.1.1 *HOT System*

We use a commercially designed HOT system from Meadowlark Optics. It is powered by an IPG Photonics Nd-YAG 5W 1064nm infrared laser that is expanded, then aimed at a 512 x 512 pixel, 100 frame per second, phase-only SLM from Boulder Nonlinear Systems. The imaging is achieved through bright-field microscopy with an oil-immersed 1.35 NA objective. We use a Teledyne DALSA HM640 which captures 640 x 480 pixel monochrome frames at a maximum rate of 300 frames per second. The Meadowlark Optics HOT System ships with a phase pattern generation software named Red Tweezers [6]. Red Tweezers implements the gratings and lenses algorithm [21] and extensions for line-based traps to compute the phase mask displayed on the SLM in real-time. An example of this is shown in Fig. 2.1., where the phase pattern and resulting backreflection of the trap is shown for the different trap types. This is only achieved through OpenGL code which runs directly on the computer's GPU. With this in mind, our control software implementation does not directly generate phase masks, but rather compiles all the current desired traps into UDP packets which are communicated to Red Tweezers and subsequently run on the GPU. Using the gratings and lenses algorithm, communicating over UDP, and moving traps has almost no perceived latency during use.

Optical Traps

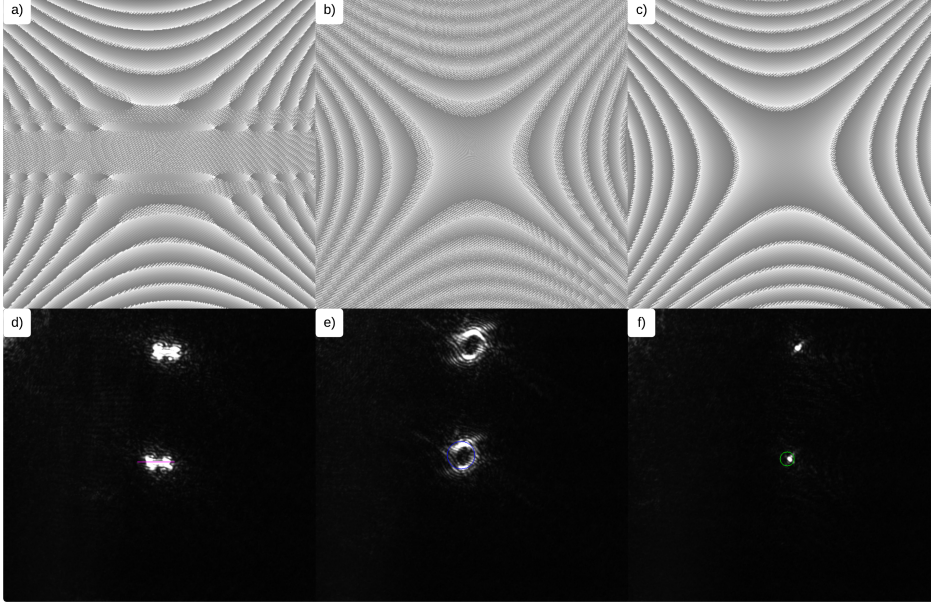


Figure 2.1: Screenshot of phase patterns and trap back reflection for the various trap types. a), b), c) are the line, annular, and point trap phase patterns respectively. d), e), f) are the imaged back reflections of the line, annular, and point traps when a mirror is placed in the sample plane.

Our SLM-based HOT system allows for control over the intensity distribution of light on workspace plane. With Red Tweezers, we can not only generate point traps, but also multiplexed traps capable of tweezing multiple polystyrene micro-beads. The three main types of optical traps we used in our experiments are described below:

- **Point Traps:** Point traps are characterized by a diffraction limited focal point with high intensity which can trap a single micro-bead and manipulate it around the workspace by simply translating the trap.
- **Annular Traps:** Annular traps are those with a topological charge applied, enabling

it to carry orbital angular momentum [15]. This type of trap is characterized by a high intensity ring which encloses a circular center point which has zero intensity. These annular traps are unique in that they can trap specific objects which generally cannot be tweezed with regular point traps due to being metallic, reflective, or absorbing. Another use for these traps is its ability to trap multiple micro-beads in a ring pattern.

- **Line Traps:** Line traps are a modified version from [33], in which arbitrary potential energy wells can be generated along 1-dimensional curves. The implementation contained in Red Tweezers can generate line traps with an arbitrary angle and length, capable of trapping multiple micro-beads in a linear fashion.

2.1.2 Problem Formulation

We formulate the HOT control problem as an optimal control problem where we wish to reach a desired goal state while avoiding collisions with other obstacles. In our experiments, the obstacles and controlled entities are polystyrene micro-spheres which are distributed throughout the microscope slide in an aqueous solution.

We define our controlled bead as agent a and our set of obstacles $O \in \mathbb{R}^2$. The goal position is G and start position is the current agent coordinates. The agent dynamics are that of a spring-mass-damper, which is an approximate model of the forces which the micro-sphere experiences while being manipulated by the HOT [36].

The continuous time state space model for the spring mass damper system is:

$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} -k/b & 0 \\ 0 & -k/b \end{bmatrix} \begin{bmatrix} x_b \\ y_b \end{bmatrix} + \begin{bmatrix} k/b & 0 \\ 0 & k/b \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

Where k is the spring coefficient in $\frac{N}{m}$, b is the damping coefficient in $\frac{Ns}{m}$, and we consider the mass m of the bead as negligible. (x_b, y_b) represent the position of the controlled bead in camera coordinates (origin centered at the top left) and (x_t, y_t) is the position of the trap.

We define our time steps as i with T denoting the terminal step, in which the bead has made its last control and reached the goal state. With this formulation, we implement a model predictive control scheme for manipulating the bead with the presence of randomly diffusing obstacles in the workspace.

Extending our problem formulation to any combination of trap type and any number of traps, we define our set of starting states as $\mathbf{S}_l, \mathbf{S}_a, \mathbf{S}_p$ which indicates line, annular, and point traps respectively. Similarly for the goal states, we define three sets $\mathbf{G}_l, \mathbf{G}_a, \mathbf{G}_p$. Our objective is to move all respective traps from their starting positions to the goal positions, while avoiding collisions with all obstacles in the set $\mathbf{O}$, as well as other concurrently controlled traps which we denote as $\mathbf{T}_l, \mathbf{T}_a, \mathbf{T}_p$. Note that the obstacle and trap positions will vary over time steps. Fig. 2.2 shows a depiction of an example workspace setup, where two line traps and one annular trap are formed and have successfully trapped beads. The obstacles, which are denoted as solid red circles, exhibit Brownian motion and move stochastically throughout the area. Goal positions are the shaded regions and are the desired goal locations for the respective traps. The objective is to manipulate the traps to their respective goal locations simultaneously while avoiding each other and obstacle beads.

2.1.3 Iterative Linear Quadratic Regulator

The iterative linear quadratic regulator (iLQR) [38] is an optimal control method which can find a locally optimal set of controls to regulate a non-linear system to a desired goal state. Utilizing running and terminal costs, the objective function can be easily tuned to include soft-constraints for region and/or obstacle avoidance. In addition, the utilization of auto-differentiation and parallelization for the forward and backward pass of the algorithm provides significant performance boosts enabling it to easily run in real-time on modern computing hardware depending on the dynamics. A summary of the overall iLQR solve process is outlined below:

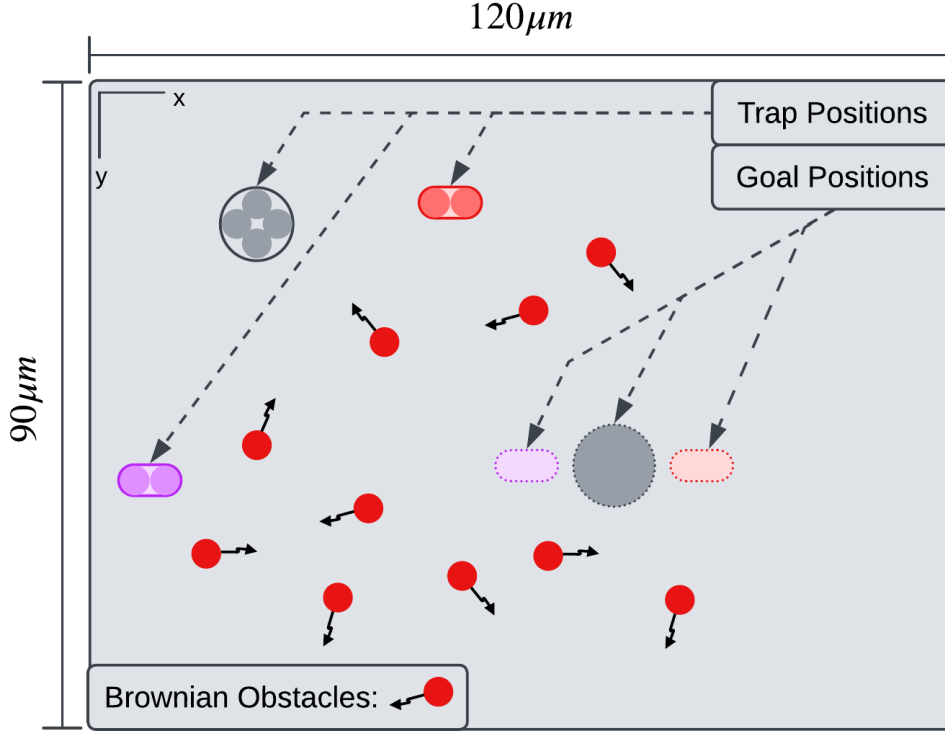


Figure 2.2: Problem formulation illustration. Red circles with jagged arrows represent freely diffusing obstacles exhibiting Brownian motion. Other shaded circles represent trapped beads. The objective is to simultaneously maneuver all trapped beads to their respective shaded goal positions without collisions.

iLQR Formulation

Considering a discrete time dynamics model of our system $\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)$, where $\mathbf{x}_i$ is our state at time step i and $\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)$ describes the state evolution of our system based on the control input $\mathbf{u}_i$. The running costs l , and terminal cost l_T are summed which results in our total cost at the initial time step J_0 .

$$J_0(\mathbf{x}, \mathbf{U}) = \sum_{i=0}^{N-1} l(\mathbf{x}_i, \mathbf{u}_i) + l_f(\mathbf{x}_N) \quad (2.1)$$

Where $\mathbf{U} = \{\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{N-1}\}$ denotes the set of controls taken from the initial time step to just before the horizon $N - 1$.

iLQR Solve

The solution to the optimal control problem is given by minimizing the cost over the entire control sequence: $\mathbf{U}^*(\mathbf{x}) = J_0(\mathbf{x}, \mathbf{U})$. The details for finding the minimizing control sequence at an initial time step are described in [38]. We describe the details for finding the optimal control sequence here for completeness.

In the context of MPC, where we find the optimal control sequence at each subsequent time step i , we define $\mathbf{U}_i = \{\mathbf{u}_i, \mathbf{u}_{i+1}, \dots, \mathbf{u}_{N-1}\}$ as the set of controls from our initial time step i . The cost-to-go is defined as:

$$J_i(\mathbf{x}, \mathbf{U}_i) = \sum_{j=i}^{N-1} l(\mathbf{x}_j, \mathbf{u}_j) + l_f(\mathbf{x}_N) \quad (2.2)$$

which can be interpreted as the sum of costs from time step i to N . We then define the value function as:

$$V(\mathbf{x}, i) = \min_{\mathbf{U}_i} J_i(\mathbf{x}, \mathbf{U}_i) \quad (2.3)$$

which is the cost-to-go given the control sequence which minimizes the cost J_i . The Dynamic Programming Principle is used to set $V(\mathbf{x}, N) = l_f(\mathbf{x}_N)$. $V(\mathbf{x}, i) = \min_{\mathbf{u}} [l(\mathbf{x}, \mathbf{u}) + V(\mathbf{f}(\mathbf{x}, \mathbf{u}), i + 1)]$. By perturbing about the i -th state-control pair:

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) = \ell(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}, i) - \ell(\mathbf{x}, \mathbf{u}, i) + V(\mathbf{f}(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}), i + 1) - V(\mathbf{f}(\mathbf{x}, \mathbf{u}), i + 1) \quad (2.4)$$

expanding this to second order, we get:

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^T \begin{bmatrix} 0 & Q_{\mathbf{x}}^T & Q_{\mathbf{u}}^T \\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}} \\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix} \quad (2.5)$$

$$Q_{\mathbf{x}} = \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^T V'_{\mathbf{x}} \quad (2.6a)$$

$$Q_{\mathbf{u}} = \ell_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^T V'_{\mathbf{x}} \quad (2.6b)$$

$$Q_{\mathbf{xx}} = \ell_{\mathbf{xx}} + \mathbf{f}_{\mathbf{x}}^T V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \mathbf{f}_{\mathbf{xx}} \quad (2.6c)$$

$$Q_{\mathbf{uu}} = \ell_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^T V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{u}} + V'_{\mathbf{x}} \mathbf{f}_{\mathbf{uu}} \quad (2.6d)$$

$$Q_{\mathbf{ux}} = \ell_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^T V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \mathbf{f}_{\mathbf{ux}} \quad (2.6e)$$

Then, we minimize the second order form in (2.5) respect to the control perturbation and obtain

$$\delta \mathbf{u}^* =_{\delta \mathbf{u}} Q(\delta \mathbf{x}, \delta \mathbf{u}) = -Q_{\mathbf{uu}}^{-1}(Q_{\mathbf{u}} + Q_{\mathbf{ux}} \delta \mathbf{x}) \quad (2.7)$$

$\mathbf{k} = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}}$ and $\mathbf{K} = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}}$ are the open-loop and feedback gain terms respectively.

The quadratized model of our Value function is now:

$$\Delta V(i) = -\frac{1}{2}Q_{\mathbf{u}}Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}} \quad (2.8a)$$

$$V_{\mathbf{x}}(i) = Q_{\mathbf{x}} - Q_{\mathbf{u}}Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}} \quad (2.8b)$$

$$V_{\mathbf{xx}}(i) = Q_{\mathbf{xx}} - Q_{\mathbf{xu}}Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}} \quad (2.8c)$$

Forward pass:

$$\hat{\mathbf{x}}(1) = \mathbf{x}(1) \quad (2.9a)$$

$$\hat{\mathbf{u}}(i) = \mathbf{u}(i) + \mathbf{k}(i) + \mathbf{K}(i)(\hat{\mathbf{x}}(i) - \mathbf{x}(i)) \quad (2.9b)$$

$$\hat{\mathbf{x}}(i+1) = \mathbf{f}(\hat{\mathbf{x}}(i), \hat{\mathbf{u}}(i)) \quad (2.9c)$$

The overall steps for a single iLQR solution is to first linearize the system dynamics $\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)$. With a locally accurate linearized dynamics model, we can then quadratize the cost function giving us a second-order approximation of the cost-to-go. The backward pass is then done, where dynamic programming is used to compute the value function $V(\mathbf{x}, i)$. The forward pass then applies the optimal control sequence $\mathbf{U}^*$ to generate an update trajectory. Iteration of both the backward pass and forward pass is done until convergence. Greater details of both the algorithm and implementation strategies can be found in [23, 38, 35]

2.1.4 Model Predictive Control

Model predictive control is implemented using the iLQR solution. The first control, or first n controls are taken from the iLQR solution and traps are moved with respect to the given control. Then the iLQR solution is recomputed given new information regarding obstacles. Our running cost function incorporates soft-constraints on movement magnitudes and obstacle separation. Movement cost C_m is calculated by taking the magnitude of the distance between our state and control, this aims to restrict the movement speed of the trap.

$$C_m = \epsilon_m (\mathbf{x} - \mathbf{u})^2 \quad (2.10)$$

The obstacle cost is given by C_o , which uses the individual distances between the state and obstacle position.

$$C_o = \sum_{k=0}^K \left(\begin{cases} 0, & \text{if separation} > \epsilon_{sep} \\ \epsilon_o (\mathbf{x} - \mathbf{O}_k)^2, & \text{otherwise} \end{cases} \right) \quad (2.11)$$

With the total running cost $C_r = C_0 + C_m$. The terminal cost is

$$C_T = \sum \left(\begin{cases} d - \epsilon_d, & \text{if distance to goal} > \epsilon_d \\ d^2, & \text{otherwise} \end{cases} \right) \quad (2.12)$$

We define a parameter c to indicate the number of control steps to execute and consequently the frequency of iLQR iterations, this is mainly changed to allow for real-time computation on the lab laptop. Each type of trap (point, annular, line) has specific parameters θ which dictate their size for obstacle avoidance and whether or not the trap is modeled as an ellipse or circle for obstacle avoidance calculations.

2.1.5 Obstacle Model

Each controlled trap avoids collisions with other traps and obstacles in the workspace by computing distances to pre-determined buffer regions. We implement two distinct distance metrics for line traps and for point/annular traps. For point and annular traps, the distance

Algorithm 1 Trap MPC Control Loop

Require: $\mathbf{O}_{\text{all}}$ (initial obstacle set), x_s (start state), g (goal state), N (prediction horizon),

$\epsilon_{\text{goal_tol}}$ (goal tolerance)

```

1:  $\mathbf{O} \leftarrow \mathbf{O}_{\text{all}} - x_s$  ▷ Exclude self start state from obstacle set
2: padObstacles() ▷ Pad obstacle array with zeros if too short
3:  $x \leftarrow x_s$ 
4:  $t \leftarrow 0$ 
5: while True do
6:   if  $t \bmod n = 0$  then ▷ Recompute iLQR solution every  $n$  steps
7:      $\text{optimal\_controls} \leftarrow \text{iLQRSolve}(x, g, \text{nominal\_control}, N)$ 
8:      $\text{ctrl\_idx} \leftarrow 0$ 
9:   else
10:     $\text{ctrl\_idx} \leftarrow \text{ctrl\_idx} + 1$  ▷ Use precomputed controls for intermediate steps
11:   end if
12:    $\text{opt\_control} \leftarrow \text{optimal\_controls}[\text{ctrl\_idx}]$ 
13:   moveTrap( $x$ ,  $\text{opt\_control}$ )
14:    $\text{nominal\_control} \leftarrow \text{opt\_control}$ 
15:    $x \leftarrow \text{opt\_control}$ 
16:    $\mathbf{O}_{\text{all}} \leftarrow \text{updateObstacles}()$ 
17:    $\mathbf{O} \leftarrow \mathbf{O}_{\text{all}} - x_s$ 
18:   removeTrappedBeads()
19:    $\text{dist\_to\_goal} \leftarrow \|x_s - g\|_2$ 
20:   if  $\text{dist\_to\_goal} \leq \epsilon_{\text{goal\_tol}}$  then
21:     break
22:   end if
23:    $t \leftarrow t + 1$ 
24: end while

```

metric is simply the distance between the center of each obstacle and the center of the trap itself. We then subtract the obstacle radius and the trap radius to obtain a set of separation distances between the controlled trap and all obstacles. Here, obstacles are defined as an augmented set $\mathbf{O}_{\text{all}} = \mathbf{O}_{\text{beads}} + \mathbf{O}_{\text{traps}}$, so that other traps are included in the total obstacle set.

Since line traps are not circular, we modeled the boundary around the trap as an ellipse. We then use a root finding algorithm to find the minimum distances from $\mathbf{O}_{\text{all}}$ to the ellipse defined around the line trap. The roots are found using an iterative bisection algorithm which guarantees a solution when searching over the domain defined by the parameters of the ellipse. Simplification of the model is done to account for multiple solutions for the algorithm as shown in Fig. 2.3. By applying a coordinate transformation to the ellipse and obstacles, we translate the ellipse to the origin and take the absolute value of all obstacle coordinates to ensure they all lie in the first quadrant, ensuring we can always find the desired distances away from the line trap boundary.

Minimum Distance from a Point to the Edge of an Ellipse

Consider an ellipse centered at the origin defined by its major and minor axis a and b , respectively, and a point with coordinates (x_p, y_p) . Our goal is to compute the minimum distance from this point to the edge of the ellipse. To proceed, we define a parameter t and use bisection on the range $[t_0, t_1]$, where:

$$t_0 = -b^2 + by$$

and

$$t_1 = -b^2 + \sqrt{a^2x^2 + b^2y^2}$$

as described in [11]. We define the function:

$$f(t) = \left(\frac{ax}{t + a^2} \right)^2 + \left(\frac{by}{t + b^2} \right)^2 - 1$$

Using bisection, we can determine the values t_{0f} and t_{1f} such that $f(t_{0f})$ and $f(t_{1f})$ are close to zero. Then compute the midpoint:

$$\bar{t} = \frac{1}{2}(t_{0f} + t_{1f})$$

To find the point on the ellipse closest to (x_p, y_p) , we calculate the coordinates (x_e, y_e) on the ellipse using $\bar{t}$:

$$x_e = \frac{a^2 x}{\bar{t} + a^2}, \quad y_e = \frac{b^2 y}{\bar{t} + b^2}$$

The distance between the point (x_p, y_p) and the closest point (x_e, y_e) on the ellipse is then:

$$d_o = \sqrt{(x_e - x_p)^2 + (y_e - y_p)^2}$$

The distance, d_o , is the minimum distance between the point (x_p, y_p) and the elliptical boundary of the line trap.

The simplifications we made to find these minimum distances do come with some drawbacks and extra considerations. Firstly, we are unable to model a rotated ellipse that does not have its principal axes aligned with the x and y axis. Second, we only know if the line trap is decreasing or increasing its distance away from obstacles and do not have a more granular view on which specific beads we are moving away from or towards. The latter does not impact the controller's actions when planning trajectories for the trap, however, it would mainly be of greater consideration for future work with higher level planners that might want to prioritize specific trajectories and move certain obstacle beads away from the line traps' path.

DBSCAN Clustering

Controlled traps often maneuver around regions densely populated with obstacles, where gaps between neighboring obstacles can be close to the diameter of the controlled trap. Our MPC implementation can sometimes get stuck in local optima, attempting to plan a trajectory through two closely spaced obstacles. To address this, we use the DBSCAN clustering algorithm to identify clusters of obstacles and place artificial obstacles between real obstacles that lie within a specified distance from each other.

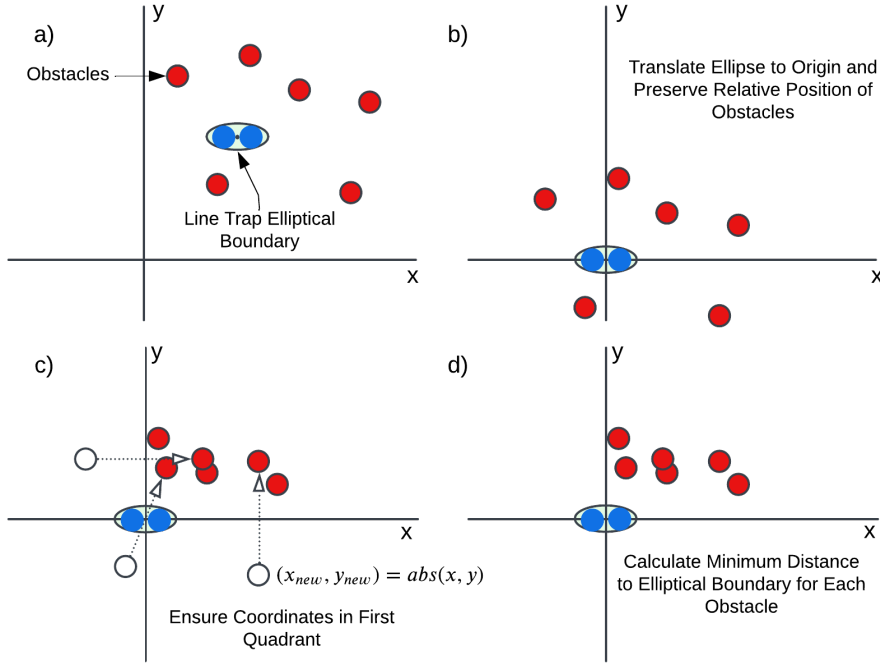


Figure 2.3: Pre-processing steps for ellipse distance algorithm. a) original state of obstacles relative to the line trap. b) coordinate system origin translated to the center point of line trap. c) absolute value of obstacle coordinates taken to ensure all stay in first quadrant. d) relative obstacle positions after pre-processing.

We utilize the DBSCAN implementation from the `sklearn` Python package, which takes an array of all obstacles as input and returns cluster labels for each obstacle. For each identified cluster, lines are defined connecting adjacent obstacles within the same cluster. If the length of a line exceeds a predefined threshold, artificial obstacle points are placed evenly along the line. This process is repeated for each line in every cluster. The artificial obstacles are then combined with the known obstacles, which include both trap locations and obstacles identified by the bead detection algorithm. The controller treats these artificial points as real obstacles, adjusting the planned path accordingly.

2.1.6 Graphical Interface

A basic graphical user interface (GUI) is implemented using OpenCV, allowing the operator to interact with a live camera feed of the workspace. Through the GUI, the operator can create, remove, move, and place goal positions for specific traps. Once the desired traps are positioned and their corresponding goals are defined, sliders are used to activate the control processes for each trap-goal combination. This setup enables the operator to plan composite trajectories, where line and/or annular traps are guided to their goal states first, followed by the maneuvering of point traps. The camera view of the GUI is shown in Fig. 2.4, where one of each trap and goal is placed in the workspace.

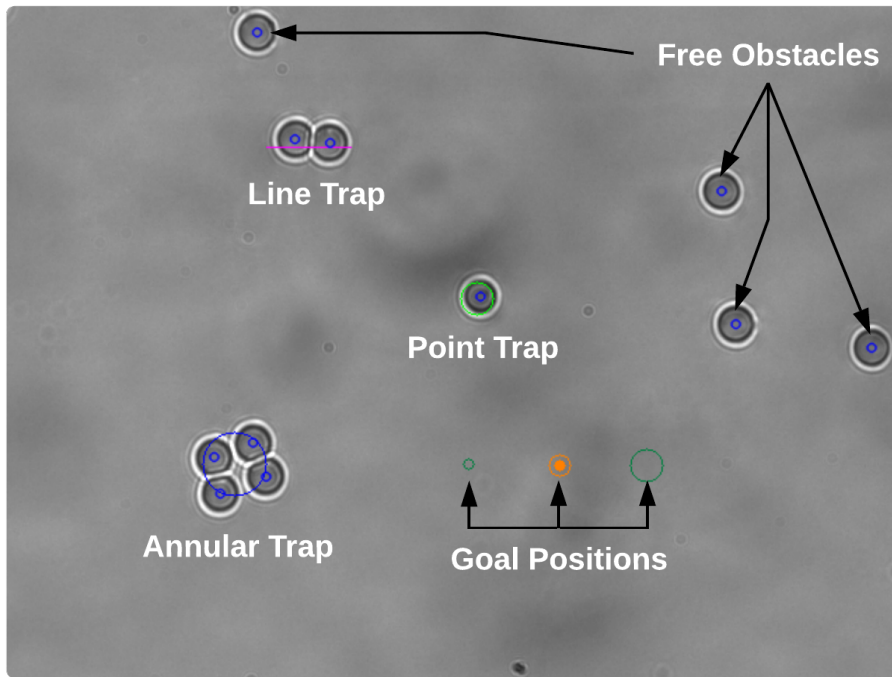


Figure 2.4: Screenshot of the optical tweezers user interface. Line traps are denoted by a purple line, annular traps with a large blue circle, and point traps with a green circle. The goal positions for the line, annular, and point trap are the small green circle, orange concentric circles, and green circle respectively.

2.1.7 Integration of Multiple Controlled Beads

During each MPC iteration, a single controlled trap is guided toward its goal. To enable the simultaneous planning and control of multiple traps, we leverage Python’s `multiprocessing` package, which supports concurrent operations through sub-processes. A custom class, `SpotManager`, manages shared data structures that define obstacles and trap locations, and provides methods to add, remove, or move traps between positions. During runtime, multiple controller subprocesses access this shared instance of `SpotManager` to receive up-to-date state information. This setup enables each controlled trap to consider the positions of other traps, allowing them to avoid potential collision paths dynamically. Although not explicitly coordinated, multiple traps can effectively avoid one another as they navigate toward closely spaced neighboring goal states.

2.2 Results

2.2.1 Implementation

A high-level overview of the implemented HOT control framework is described in Fig. 2.5. Detection of the polystyrene micro-beads used in our experiments is achieved using the OpenCV blob detector class as outlined in [28]. We use a modified version of the iLQR implementation in [35] for our control framework.

Trap Intensity Scaling

Red Tweezers does not automatically adjust the intensity of each trap to ensure that the average trapping intensity per individual trap remains constant when traps are added or removed. Due to the minimum laser power output limitations, we cannot modulate the laser power below 0.9 W, which is relatively strong for a single point trap. Creating a trap at this power level results in a robust optical trap with a low probability of losing the trapped micro-bead. However, this also introduces challenges, as freely diffusing neighboring micro-beads are drawn to the strong potential, increasing the likelihood of unintended trapping. This high trapping power prevents positioning the trap close to other obstacles without the risk of unintentionally trapping them, which leads to collisions and potential optical binding

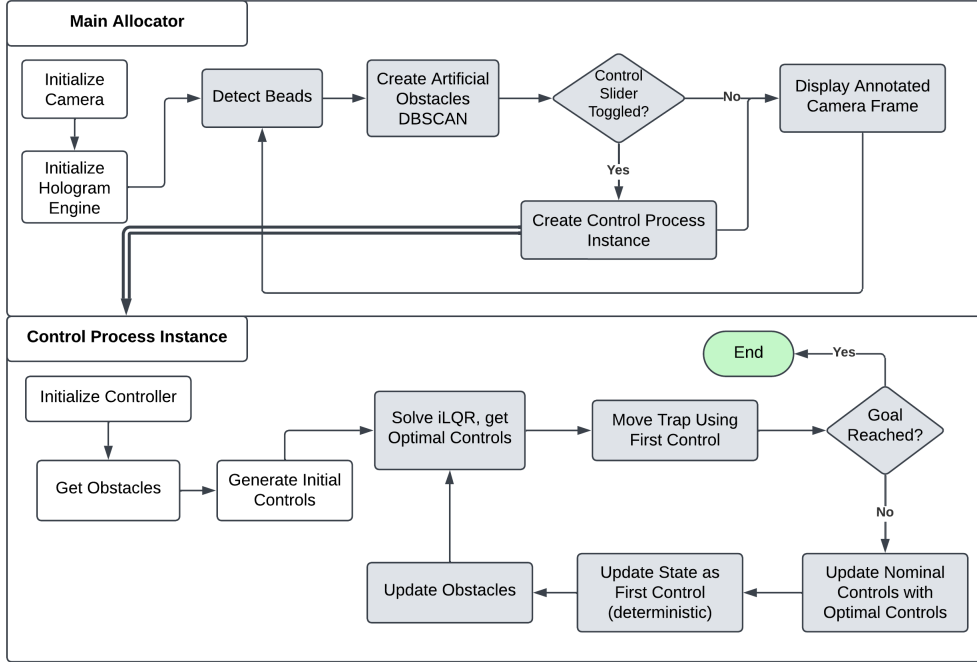


Figure 2.5: Overview of the HOT automation framework. The main allocator is run on startup and maintains the state of the entire workspace. Once the operator has traps and goals placed, sliders are toggled to start a separate control process for each trap-goal combination.

of beads [13].

To address this, we implemented a trap power modulation strategy within the tweezers control framework. This adjustment generates additional traps near the zeroth-order spot of the holographic optical trap. With this configuration, if only a single trap is required, additional traps are generated to maintain a relatively constant power level between scenarios with one and multiple traps. As the operator adds traps to the workspace, these artificial traps are sequentially removed until a certain threshold is reached. Conversely, as traps are removed from the workspace, the number of artificial traps increases until only a single trap remains in the workspace.

We introduce a basic formulation for this power scaling to mitigate the issues described above:

Algorithm 2 Trap Power Scaling Algorithm

Require: n_{traps} , $n_{\text{virtual_traps}}$, laser_power , avg_desired_power

```

1:  $P_{\text{total}} \leftarrow \text{avg\_desired\_power} \times (n_{\text{traps}} + n_{\text{virtual\_traps}})$ 
2: if  $P_{\text{total}} > \text{laser\_power}$  then
3:    $n_{\text{remove}} \leftarrow \left\lfloor \frac{P_{\text{total}} - \text{laser\_power}}{\text{avg\_desired\_power}} \right\rfloor$ 
4:   if  $n_{\text{virtual\_traps}} > 0$  then
5:      $\text{removeVirtualTraps}(n_{\text{remove}})$ 
6:   end if
7: end if

```

JAX Parallelization

Both the iLQR and ellipse distance finding algorithm are implemented using JAX, a Python package which enables numerical computations to be accelerated by pre-compiling code written in Python into XLA then performing the desired calculations. In addition, JAX provides useful tools for auto-differentiation and array-based batch operations which makes it desirable in our application. The use of JAX, however, requires careful consideration of what types of inputs are passed to functions which have been pre-compiled in XLA. Acceleration of certain calculations, such as the forward pass of iLQR can be achieved under condition that it is a pure function and that array sizes are constant. This became prevalent during runtime of earlier versions of the MPC framework, where the array containing states of the obstacles did not have a constant size, leading to recompilation of the Python code during runtime causing stutters and performance degradation. A rather straightforward solution was devised where the obstacle array was padded with zeros and then passed into JAX compiled functions. This alleviated the recompilation issue but leaves the designer to choose the array size accordingly. A large obstacle array ensures that the program will not likely run into size constraint issue but it also leads to increased computational load due to the separation distance calculations. This consideration is especially important when dealing with line traps which have elliptical boundaries since separation distance is calculated using the iterative bisection method.

MPC Parameters

Many of the parameters described in our MPC implementation are with obstacle buffers, running costs, and terminal costs. Table 2.1 contain the experiment parameters we used for both the dynamics model and objective function. Table 2.2 contains differing parameters for each trap type, which is passed to the iLQR solver before initialization. The trap specific parameters mainly serve to distinguish obstacle buffer sizes and the obstacle model to use. Other trap specific parameters can also be specified if desired, such as those defined in Table 2.1. Note that all distance measurements are given in pixels (px). The spring and damping constants are given in $\frac{N}{m}$ and $\frac{Ns}{m}$ respectively. In addition, the nominal trajectory used for initializing iLQR was an evenly sampled set of states for the line segment connecting the starting state and goal state.

Table 2.1: Experiment Parameters

Parameter	Value	Parameter	Value
ϵ_o obstacle cost multiplier	10^2	b damping coefficient	10^{-8}
ϵ_d goal distance activation	25px	k spring constant	10^{-5}
ϵ_m movement cost multiplier	10^3	n iLQR computation interval	2
a_{minor}	12px	b_{major}	30px

Table 2.2: Trap Type Specific Parameters

Parameter	Annular Trap	Line Trap	Point Trap
ϵ_{sep}	1.2 * 32 px	32px	32px
agentAsEllipse	False	True	False
ellipseAxis	None	(a,b)	None

2.2.2 *Experimental Setup*

We conduct experiments using the HOT system by using commercially produced polystyrene micro-bead concentrate, diluting it into a solution with DI water, then agitating it using a vibratory shaker. The $30\mu L$ solution is then quickly transferred via pipette to a microscope slide and sandwiched with a slide cover. The polystyrene micro-beads used in our experiments are from Bangs Laboratories and are $5\mu m$ in diameter. A solution containing 0.074% micro-beads by volume enables us to work with a reasonably dense, obstacle-rich environment and allows additional beads to be used for the controlled traps.

2.2.3 *Experiment Execution*

The experiments are conducted by an operator interacting with the OpenCV GUI to create various traps, assemble micro-beads into the traps and set goal positions. The experiment procedure is as follows:

1. **Trap Generation:** Referencing a desired pattern to assemble, the operator places all required traps onto the workspace. The trap is then moved to capture bead(s) and fully populate the trap.
2. **Set Goal Positions:** Goal positions are set for the different types of traps using keyboard shortcuts and right-clicking the desired goal position on the camera feed. The goal positions correspond to the pattern we wish to assemble and the ordering of traps to goal placements are equal (e.g. the second line trap we place will correspond to the second line goal placed)
3. **Start Controllers:** The sliders are toggled and sub-processes are started for each trap and goal combination. The traps will be controlled to their desired goal location while considering other traps and detected obstacle beads in the workspace.
4. **Wait Until Goal State Reached:** Once the traps have started, the operator waits until all traps are within the goal position threshold and documents the experiment.

Alternatively, if traps lose beads while being controlled, or collisions occur, the trial is marked as a failure.

2.2.4 Experiments

In order to test the performance of the MPC control framework, we devised a set of classes for formations which created our total set of 30 formations tested. These formations contained two or more types of traps and two or more traps being assembled into the desired pattern. The general classes are outlined in Fig. 2.6.

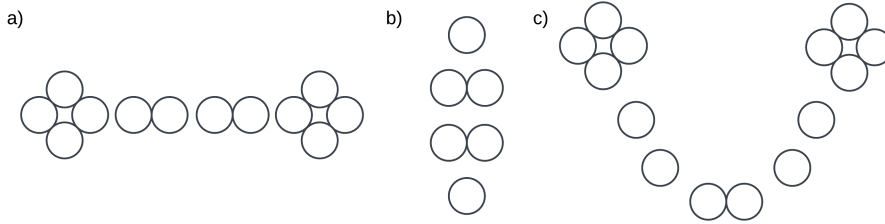


Figure 2.6: Depiction of the various classes of bead formations assembled for experiments. a) is a formation comprised of both line and annular traps in a dumbbell-like pattern. b) consists of point and line traps assembled to represent a diamond. c) contains all three types of traps and forms a parabolic arc.

2.2.5 Experimental Results

The measured summary statistics include the number of unintentionally trapped beads within the workspace, the total experiment duration, and the total number of beads successfully transported. Overall, the experiments demonstrate effective control, with the traps successfully avoiding collisions with obstacles and other traps during transport. Representative experimental runs are illustrated in Fig. 2.7, Fig. 2.8, and Fig. 2.9.

In most cases, the trap trajectories closely follow the initial planned trajectory, with minor deviations primarily to circumvent obstacles or other moving traps. The trap speed varies dynamically along the trajectory, slowing down significantly near obstacles or when approaching the goal state. Fig. 2.7 illustrates an experiment involving all three trap

types moving sequentially toward their respective goal states. The traps effectively avoid collisions, with point traps slowing down and waiting for the goal region to clear before proceeding.

Fig. 2.8 depicts an experiment with line and point traps, where one of the line traps takes a longer path to avoid a moving point trap. Finally, Fig. 2.9 showcases an experiment with annular and line traps, where the line trap navigates around dynamic obstacles moving toward the goal states. The line trap successfully maneuvers through a gap between the obstacle and the annular trap to reach its goal.

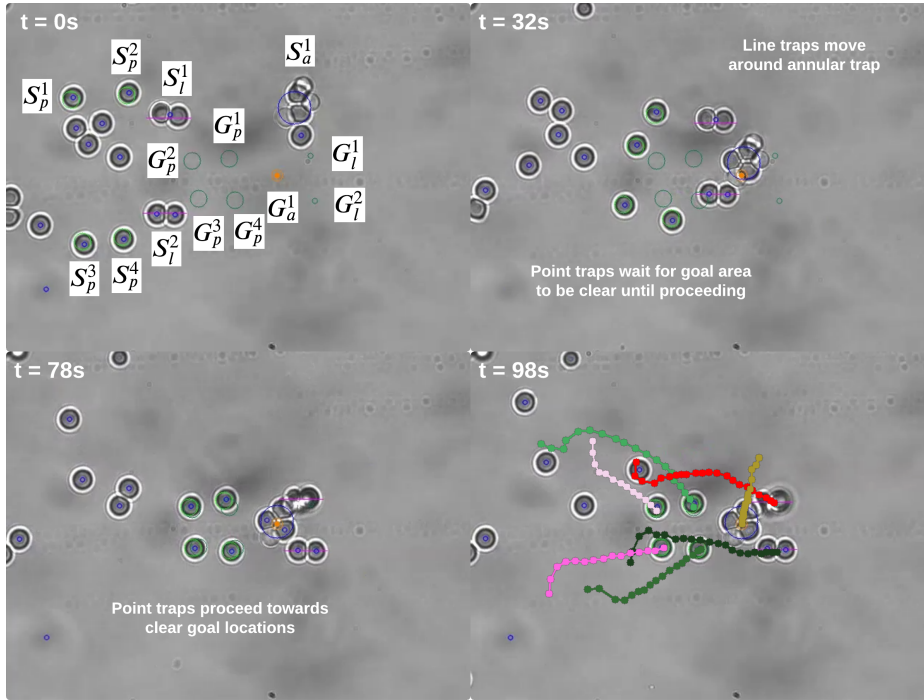


Figure 2.7: Experiment with annular, line, and point traps. Starting and goal positions for each trap type are given at $t=0s$. $t=98s$ shows the end of the experiment along with the overall trajectory for each trap.

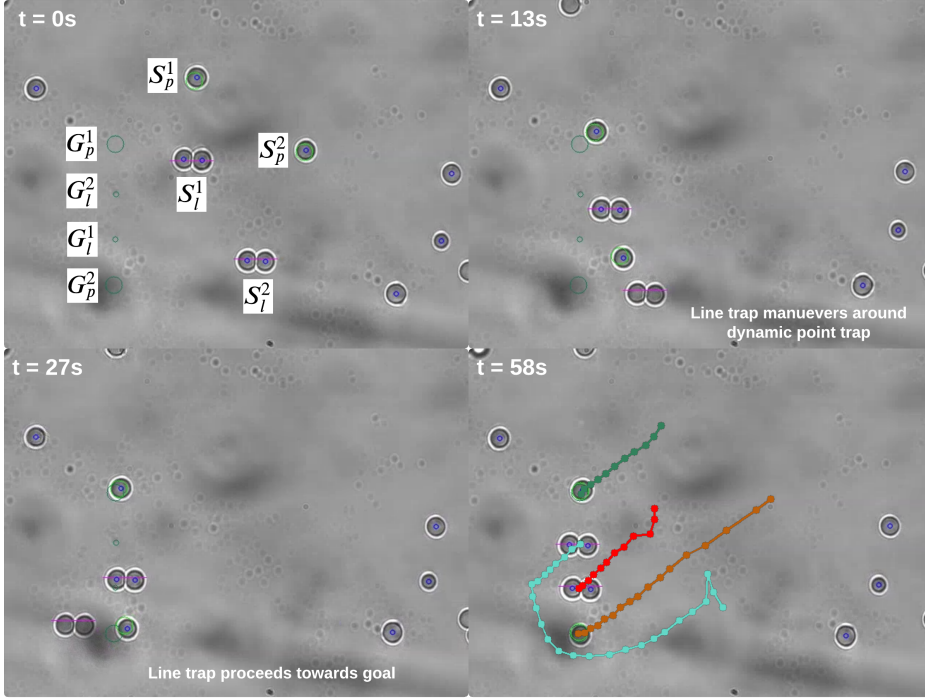


Figure 2.8: Experiment with line and point traps. Starting and goal positions for each trap type are given at $t=0s$. $t=58s$ shows the end of the experiment along with the overall trajectory for each trap.

Table 2.3: Summary of Experiment Results

Metric	Extra Trapped Beads	Duration (s)	Beads Transported
Average	0.90	55.63	9.17
Std Dev	1.06	14.85	3.54

2.3 Discussion

Integrating a model predictive control scheme with an iLQR solver is a powerful method when guiding single or multiple traps. In crowded environments, our controller was able to avoid the obstacles and actively move away from approaching obstacles. Performance with the various types of traps differed greatly, however, as both the multiplexed traps oftentimes

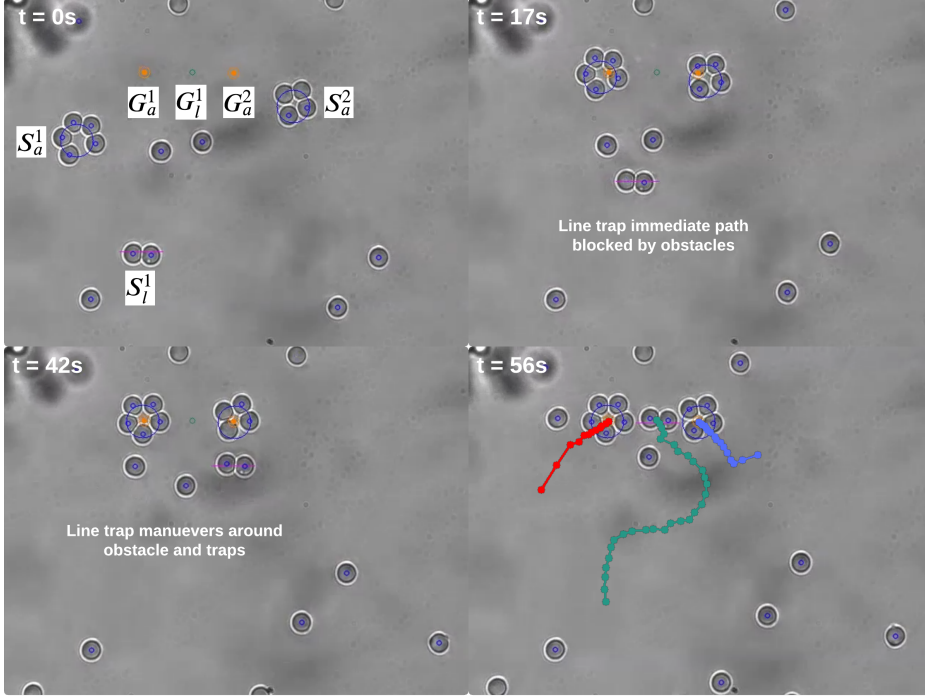


Figure 2.9: Experiment with annular and line traps. Starting and goal positions for each trap type are given at $t=0s$. $t=56s$ shows the end of the experiment along with the overall trajectory for each trap.

struggled to successfully keep the micro-beads in the desired pattern. In many instances, the annular traps would start translating and only a portion of trapped beads would follow, indicating the trap intensity being too low. By tuning the intensity for annular and line traps, we were able to achieve a balance of increasing trap power with an increased chance of unintentional trapping of nearby obstacles. At times the stronger and larger annular traps tended to attract nearby obstacles and the controller would react by moving away from the incoming obstacle. At times, the controlled trap would not be able to escape the obstacle and the trap would be controlled out of bounds, or the trap movement speed would exceed the limits which kept the beads inside of the trap.

In our previous work, [44], we used a potential-field based planner which discretized the workspace into a coarse grid in which trap trajectories would be planned once and

executed. Without any active replanning or consideration for the freely diffusing obstacles, it is challenging to achieve successful runs because of the dynamic nature of the workspace. Actively trapping all of the obstacles improved the performance of the potential-field based method, however, placing a large quantity of traps reduced the overall trapping power for the controlled traps.

Using DBSCAN to group beads with small separation distances was helpful in preventing the issue of the iLQR solver becoming stuck in a state of local minima. In environments with little crowding, it successfully identified areas in which those local minima would occur and placed artificial obstacles to mitigate this issue. Alternatively, in high crowding environments greater than around 30 obstacles, many artificial obstacles would be placed due to how coordinates of a cluster are extracted from those identified. This could be cause for concern since the array which contains the set of obstacles has a fixed size chosen with regards to the expected number of real obstacles in the scene. The artificial obstacles approach does not scale well when dealing with dense environments.

Fig. 2.10 shows an interesting two-stage experiment where three line traps formed a stair-step configuration, afterwards two point traps were placed between the gaps of the line traps. During the first stage, all three line traps were controlled towards their respective goal states and by $t = 39s$, two out of three of the line traps had nearly reached their goal. However, the second line trap, with starting state S_l^2 could not reach its goal state because the path was almost completely closed off by obstacles. Without any human intervention, the line trap loitered close to some obstacles until the beads diffused to give enough clearance for the line trap to continue at $t = 117s$, more than a minute after the first two line traps reached their goal states.

As we work towards greater autonomy capabilities, it's important to consider some of the limitations of the current control framework. As we use a fixed time horizon which we optimize over, the controlled trap has a natural tendency to slow down as it approaches the goal as control sequence returned by iLQR is spaced closer together. In future work, different dynamics models can be used to further tune characteristics of the controller, such as imposing limitations on trap acceleration or velocity. This could help reduce the likelihood of the trap loosing its micro-beads as soon as it starts translating. In our current

implementation, the desired speed of the trap can be indirectly modified by changing the movement penalty, or alternatively increasing the time horizon N , but also leads to performance degradation due to increased computation. Additionally, obstacle-trap distance calculations for the ellipse model are longer than that for point and annular traps due to the use of the bisection root finding algorithm and requiring coordinate transformations. A method which works an arbitrarily oriented line trap could enable a non-linear dynamics model, such as that of a unicycle, to rotate the line trap to slip through tighter gaps between obstacles as well as construct formations requiring line traps which are angled. The use of iLQR, usually intended for use with non-linear models, can then be repurposed easily for any non-linear model.

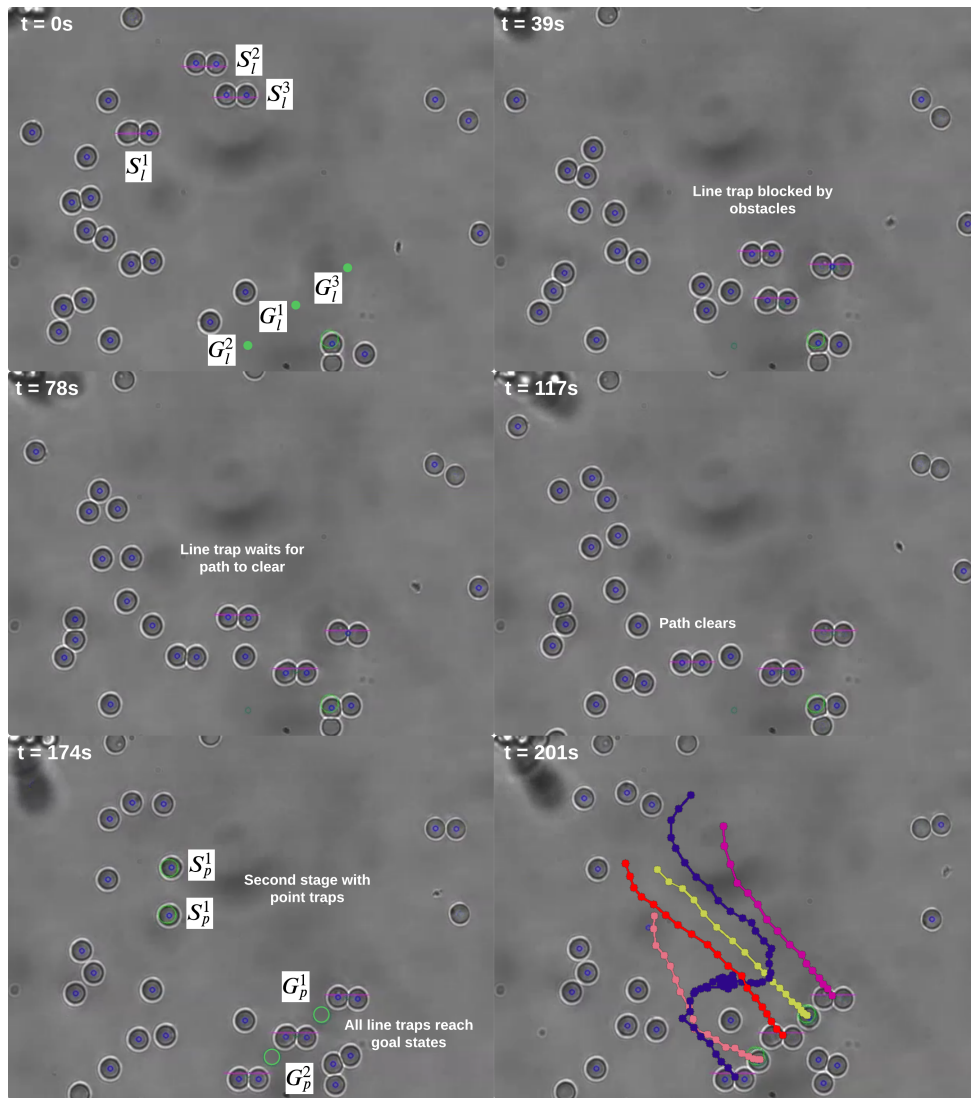


Figure 2.10: Stair step pattern formation. The pattern was assembled in two stages, the first being all three line traps and second being the two point traps.

Chapter 3

DEEP LEARNING BASED HOLOGRAM GENERATION

After the development of the MPC controller, it was evident that greater autonomy of HOTs would require more advanced strategies and a higher degree of feedback.

Trapped beads could become free during movement of the trap from the MPC controller. This can happen even at low movement speeds, especially with annular and line traps which do not have pre-defined locations for each bead trapped. For example, polystyrene micro-beads in an annular trap are rotated around the center of the trap due to the orbital angular momentum of light. In addition, line traps can have multiple points along the line which the micro-beads can be trapped, but during movement of the trap the micro-beads can displace to other points within the line trap. This makes accurately positioning individual positions of micro-beads using the complex traps challenging. In comparison, a point trap is defined by a single diffraction-limited focal point of light and a micro-bead can only be trapped at that point. These complex trap types available for use in Red Tweezers are powerful and enable a high volume of micro-beads to be transported in a single trap. However, Red Tweezers is inherently restricted to the implemented trap types and cannot be extended to create an arbitrary trap. Here, an arbitrary trap would be defined by a general light intensity distribution at the sample plane. Arbitrary light intensity distributions could be leveraged by neural network models for the full control stack (obstacle avoidance, goal seeking, advanced maneuvers such as optical binding mitigation).

Arbitrary intensity distributions at the sample plane can already be created through the use of the Gerchberg-Saxton algorithm but computation requirements for 512×512 phase masks can quickly limit the rate at which the distributions can update. As frame-by-frame changes are how optical traps with micro-objects move around the workspace, the slow calculation speed of the GS algorithm makes it undesirable for this task.

The U-Net, a commonly used fully convolutional neural network architecture, is usually

used for image segmentation where a region of interest is extracted from some input image. U-Nets are known to perform well with spatially structured data and can also be used for regression tasks where the input and output are the same dimensionality.

With the potential benefits of a general phase retriever like the GS algorithm and the GPU accelerated nature of neural networks, we were motivated to explore the U-Net architecture in acting in-place of Red Tweezers and GS algorithm for phase retrieval in our HOT system.

In this chapter we will cover the implementation strategies and training pipeline for using a deep learning model architecture to act as a phase retriever in HOT applications to directly manipulate micro-objects using a general intensity-based input distribution.

3.1 Methods

3.1.1 Model Architecture

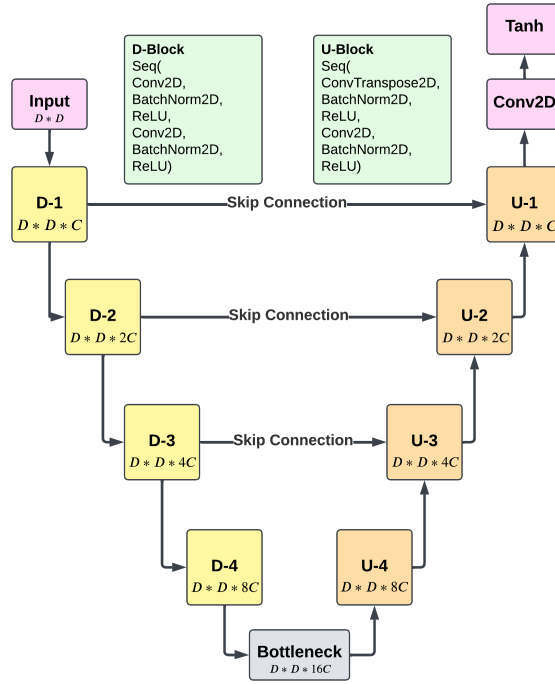


Figure 3.1: U-Net architecture for phase mask generation in holographic optical tweezers

CNNs are widely used for imaging tasks, such as image segmentation, classification, and other specialized applications [34]. The U-Net architecture that performs best with 256×256 intensity and phase arrays is illustrated in Fig. 3.1. The U-Net model comprises encoder and decoder blocks. The encoder block consists of a sequence of two convolutional layers, each followed by batch normalization and ReLU activation. The decoder block is similar, except that the first convolution is replaced by a transposed convolution. Additionally, the model is fully skip-connected, meaning that each subsequent encoder-decoder layer is concatenated during both training and inference. The output is then fed into a tanh layer that restricts the output values between 0 and 1. In the encoder layers, the feature dimensionality doubles at each step, trading spatial resolution for feature resolution. Similarly to auto-encoder approaches, the U-Net learns the salient features relevant to both the input and output during training. The decoder layers then use these salient features for prediction, while the skip connections help the model retain the finer details that can otherwise be lost in the final *D4* layer.

3.1.2 Training Data

Intensity distributions and phase masks are used as input and output of the CNN, respectively. A scalable approach is essential, as the number of training samples required for good performance is unknown at the outset. The ability to generate an arbitrary number of these samples allows for quick scalability of the model to higher dimensionality. To address this issue, we developed our own end-to-end workflow in Python using the Red Tweezers algorithm [6] to generate various optical traps and the methods in the MATLAB `otslm` toolbox [22] to simulate the intensity of light at the focal plane.

The Red Tweezers OpenGL Shading Language (GLSL) implementation is translated into Python with a similar degree of parallelization using JAX, a high-performance computing toolbox. The Red Tweezers port accepts an array of parameters that indicate the properties of the optical traps. Each individual trap is parameterized by a matrix comprising 13 individual parameters, termed as the `spot_matrix`, which encode the position, type, and special properties of the trap. A single call to Red Tweezers contains an array consisting

of all the sets of trap parameters. The output is a phase mask intended for projection onto an SLM to yield the desired optical traps.

$$\text{spot.matrix} = \begin{bmatrix} x & y & z & l \\ I & p & - & - \\ na_x & na_y & na_r & - \\ l_x & l_y & l_z & l_p \end{bmatrix}$$

- x, y, z : Coordinate of trap center in μm
- l : Laguerre–Gaussian coefficient
- I : Intensity
- p : Phase
- na_x, na_y : Numerical aperture (NA) direction components
- na_r : NA radius
- l_x, l_y, l_z : Line lengths in direction
- l_p : Line trap phase

The far field intensity of the phase mask is calculated by using the Fourier transform to shift between the two image planes. We also simulate the effect of the incident laser beam onto the SLM by using a Gaussian distribution. Using our Python implementation of the Red Tweezers algorithm, we then generate N pairs of samples that contain the simulated far field distribution as the input and the phase mask that produced the intensity distribution as the output. Note that the far field distribution includes more than the desired workspace, with the incident beam and zeroth-order lobes at the center of the simulated intensity distribution. While we are only interested in one quadrant of the intensity distribution,

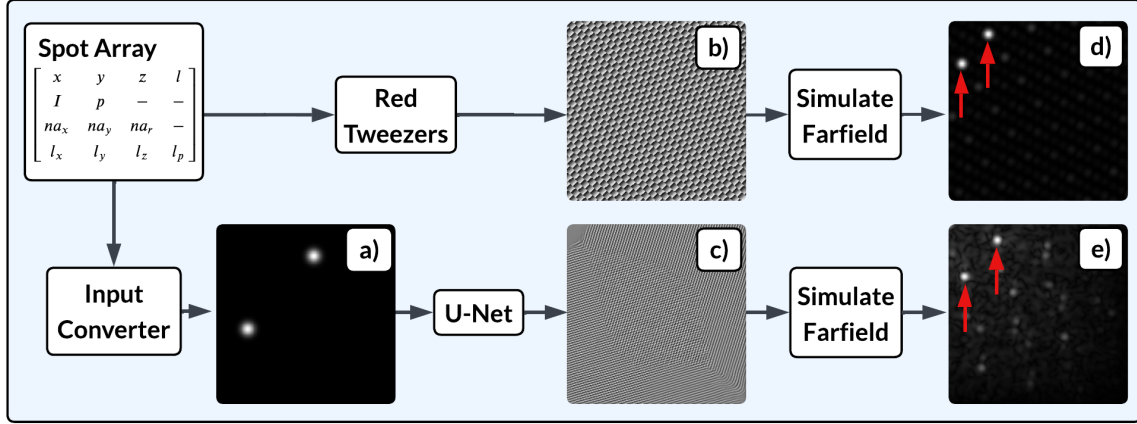


Figure 3.2: Flowchart highlighting the difference between the U-Net and Red Tweezers phase mask calculation approaches. a) Input intensity distribution to U-Net model indicating placement of two point traps. b) Phase mask calculated using Red Tweezers. c) Phase mask prediction from U-Net model given a) as input. d) Simulated far field of b), note the presence of additional orders of diffraction. e) Simulated far field of c) indicating the correct locations of the two point traps. Red arrows indicate the corresponding point traps shown in a)

better results are obtained by retaining the full field of view during training. The phase mask and intensity distribution pairs are $D \times D$ pixels and are both normalized. The phase mask is normalized between -1 and 1 and the intensity distribution is normalized between 0 and 1 . The generation of samples is parallelized using Python's `multiprocessing` package that allows multiple CPU cores to generate portions of the entire dataset.

3.1.3 Technical Approach

We formulate the problem as a phase prediction problem, where a desired intensity distribution is given that contains information about the location and types of optical traps, and the objective is to retrieve the phase mask corresponding to the intensity distribution. The data generation pipeline requires consideration of some physical calibration factors, such as the Zernike coefficients for Red Tweezers. These parameters directly impact the phase mask produced by Red Tweezers and assuming a perfect optical system results in distorted intensity distributions. Therefore, we require two phase masks, one being a reference mask

with no calibration parameters assuming an ideal system, and another mask containing the calibration parameters for our specific HOT system. By simulating the reference phase mask's far field intensity distribution, we obtain the ideal intensity distributions for training the model. Following the approach in the `otslm` MATLAB package, we use a Gaussian incident beam to model the laser: $g(r) = \exp\left(\frac{-r^2}{2\sigma^2}\right)$. We then construct a complex field from the real-valued phase mask p : $c = \exp(i * p)$, based on which we calculate the modulated complex field with the incident beam as $S = cg(r)$. The overall process to simulate the far-field intensity of the modulated phase mask is described in Algorithm 1.

Algorithm 3 Simulate Far-Field Intensity

- 1: **Input:** `phase_mask.reference`, mask dimension `D`, and beam width `bw`
- 2: Generate a Gaussian beam:

$$A(x, y) \leftarrow \text{gaussian_beam}(D, bw)$$

- 3: Create a phase-modulated field:

$$E_{\text{SLM}}(x, y) \leftarrow A(x, y) \cdot e^{i \cdot \phi(x, y)}$$

- 4: Compute far field via 2D Fast Fourier Transform (FFT):

$$E_{\text{far}}(u, v) \leftarrow \frac{\text{fftshift}(\text{fft2}(E_{\text{SLM}}))}{(2 \cdot D)^2}$$

- 5: Calculate intensity:

$$I(u, v) \leftarrow |E_{\text{far}}(u, v)|^2$$

- 6: **Output:** Intensity distribution $I(u, v)$
-

The process described in Algorithm 1 is used to generate the intensity distributions for both the training samples and the input converter. The input converter generates similar intensity masks as those in the training set, except that they are computed using the saved intensity distributions for the individual trap types. The Red Tweezers algorithm is used to generate a phase mask for a reference trap, which is then saved. During execution, we simply combine all of the desired trap's intensity distributions into a single mask to feed

Table 3.1: Model and Training Parameters

Description	Parameter	Value
Input/Output Dimension	D	256
Training Samples	N	200,000
Maximum Number of Spots	n_p	15
Learning Rate	l_r	0.0001
Base Feature Dimension	C	64
Kernel Size	k	5

into the U-Net model.

3.1.4 Execution

The model accepts inputs via an OpenCV user interface that displays a live view of the sample plane along with toggles to control the trap type. A trap is added at the location clicked by the user. Each trap that gets added, removed, or moved requires updating the desired intensity distribution fed into the U-Net model. The input converter parses the trap parameters and places the saved copies of the intensity distributions for each trap in its corresponding location. The U-Net then predicts the phase mask that is displayed onto the SLM.

3.2 Experimental Evaluation

3.2.1 Experimental Setup

We train the U-Net with the parameters described in Table 3.1 and achieve a validation mean squared error (MSE) of 0.292 after 30 epochs. This took three days when trained with two Nvidia GTX 1080Ti machines. Each sample in the training and validation dataset is generated by choosing a randomly generated `spot_matrix`. For our experiments, we only choose random parameters for x, y, l, l_y , which allows us to obtain random samples of point traps, line traps, and annular traps throughout the workspace. The location of the trap

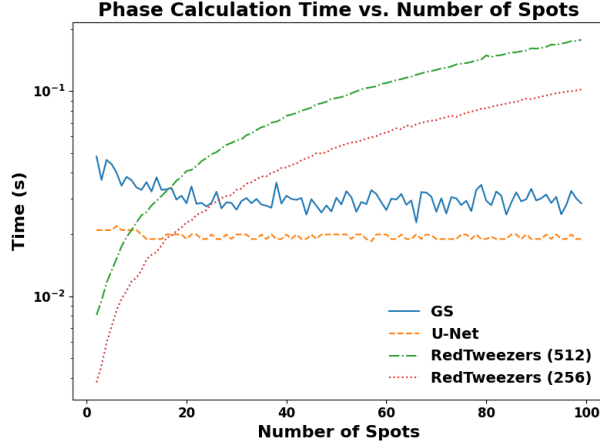


Figure 3.3: Phase mask calculation time comparison between Red Tweezers, Gerchberg-Saxton (GS), and U-Net methods.

is chosen by sampling uniformly between 0 and 120, which is the maximum dimension of the workspace in μm . The probability of a sample being a point trap, line trap, and an annular trap is chosen to be 0.8, 0.1, and 0.1, respectively. We use this trained model for the subsequent experiments that measure the mask generation (computation) time and evaluate its effectiveness on our HOT system for micro-bead movements.

For our experiments, we use a commercially available kit developed by Meadowlark Optics (Frederick, CO, USA) [6]. Imaging is performed using a Teledyne DALSA (Waterloo, ON, Canada) HM640 monochrome camera, which captures 640×480 pixel frames at up to 300 frames per second. The optical system includes a 1.35 NA oil-immersion objective, resulting in an effective magnification of $40\times$. This yields a field of view measuring approximately $120 \mu m \times 90 \mu m$, which can be translated in both lateral directions via an electronically controlled microscope stage. Phase modulation is achieved using a 512×512 pixel, phase-only spatial light modulator (SLM) from Boulder Nonlinear Systems, operating at a refresh rate of 100 Hz. Optical trapping is powered by a 5 W, 1064 nm Nd:YAG laser from IPG Photonics (Oxford, MA, USA), with adjustable output power.

3.2.2 Results

The Red Tweezers algorithm using the compiled XLA acceleration package JAX is capable of computing the phase masks in real-time. However, the computation time scales with the number of desired traps, since the phase is calculated for each pixel over each trap, whereas the GS and U-Net methods do not directly scale with the number of traps desired. Fig. 3.3 shows the comparison among all the three phase retrieval methods. After running each method with a varying number of traps, we observe that the U-Net method is faster than both the GS and the Red Tweezers algorithms. The GS algorithm is iterative, implying that we terminate the process when the error between the target intensity and the simulated intensity is below a threshold. However, in many cases, the process continues until a maximum iteration counter is reached, which results in an unsatisfactory phase mask. We timed the Red Tweezers algorithm on both 512×512 and 256×256 output dimensionality for the sake of completeness.

In order to use the U-Net model on our HOT system, we first must match the dimensionality of the phase-mask prediction to that of the SLM on the HOT system. Due to limited computational resources, training a U-Net on the full SLM resolution of 512×512 is infeasible. However, we still use the U-Net trained on 256×256 samples by upscaling the phase mask using nearest neighbor interpolation. This enables operation of the HOT system with a reduced workspace area. The workspace area is one half of the original, as the maximum deflection angle of the SLM is reduced due to upscaling. Fig. 3.4 shows the comparison between the reference phase mask calculated using Red Tweezers and the U-Net prediction for a single annular trap placed at the center of the workspace. The two phase masks are qualitatively similar; the center sections (corresponding to the annular trap) of the far-field intensity distributions are also similar, although the one from the U-Net model is fainter. Fig. 3.5 shows the U-Net predictions for a single line trap moved to various locations in the workspace. As the predicted phase masks change, the corresponding far-field intensity distributions vary accordingly, mimicking the changing location of the line trap.

We also performed proof-of-concept functional micro-object trapping and manipulation tests on our HOT system using point, line and annular traps generated by the 256×256

U-Net models. As a representative example, Fig. 3.6 shows successful maneuvering of two $5\mu\text{m}$ diameter polystyrene beads using a line trap in a moderately crowded workspace.

3.3 Discussion

The ability of the U-Net to act as a phase retrieval mechanism for given intensity distributions is powerful. Our proposed methodology of combining the saved intensity distributions from the simulated far-field intensity is effective at reproducing the trap configurations encountered during training and showed successful trapping when demonstrated on our physical HOT system. We specifically considered point, line and annular configurations, of which the latter two could be used to tweeze multiple micro-objects concurrently using a single trap [45].

Our U-Net model could lead to other strategies to manipulate and control micro-objects in the HOT workspace. One exciting potential is the ability to input an arbitrary intensity distribution. Although our model was not explicitly trained with general intensity distributions, it was able to reproduce the characteristics of arbitrary intensity distributions, as shown in Fig. 3.7. In this case, an input intensity is drawn by a human. The far-field intensity simulation shows a fainter reproduction of the drawn intensity distribution. This indicates that the model has learned suitable features to reproduce general intensity distributions instead of only memorizing the trap configurations seen in the training set.

One important consideration in our methodology is the implementation of correction parameters, such as Zernike coefficients, into the Red Tweezers phase mask calculator. This motivates our choice of using an ideal phase mask to simulate the far-field intensity input and the corrected phase mask for the U-Net output during training. As the higher dimensionality U-Net models ($D = 256$) take an extended period of time to train, we use the easier to train U-Net model with $D = 128$ and kernel size $k = 3$ to generate the annular trap in Fig. 3.4 e) and f) and the arbitrarily shaped traps in Fig. 3.7. However, this issue could be potentially addressed by modifying the distributions of point, line, and annular traps in the training dataset.

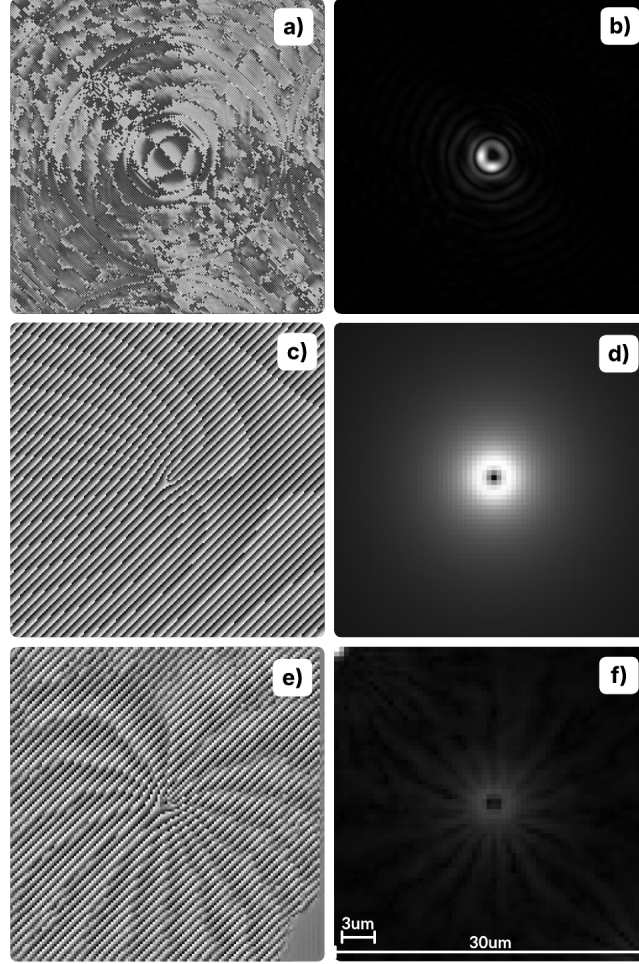


Figure 3.4: Comparisons for a single annular trap. a) Gerchberg-Saxton (GS) generated phase mask b) Far-field simulation of a). c) Red tweezers generated phase mask. d) Far-field simulation of c). e) U-Net predicted phase mask (128×128 model). f) Far-field simulation of e)

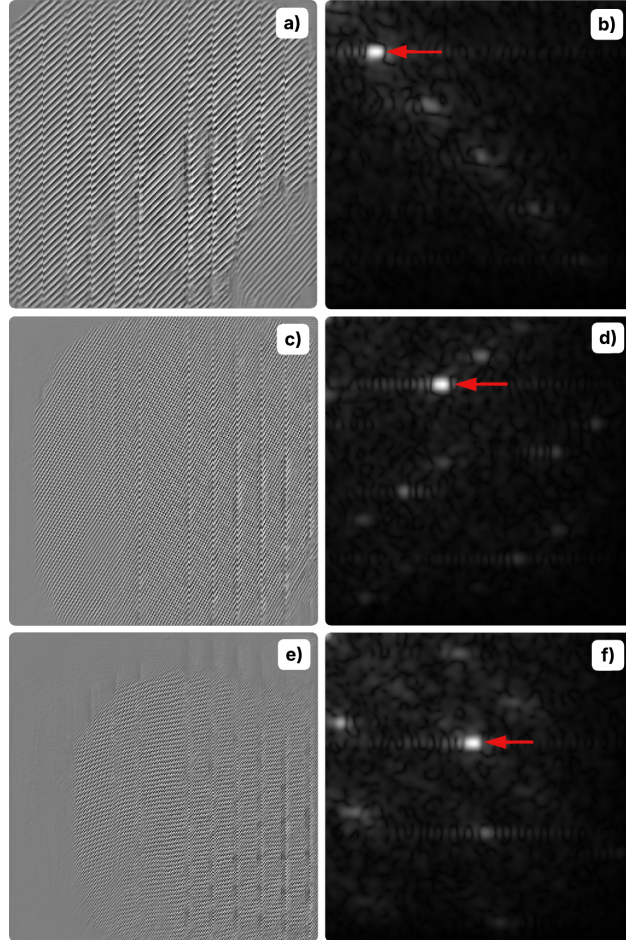


Figure 3.5: Phase mask predictions for a moving line trap. a) U-Net predicted phase mask. b) Far-field simulation of a) with the red arrow indicating the correctly reproduced location of the line trap. c) and d), e) and f) are pairs of U-Net predictions and simulated far-fields of the single line trap in different locations. The red arrows indicate the region of interest containing the reproduced line trap.

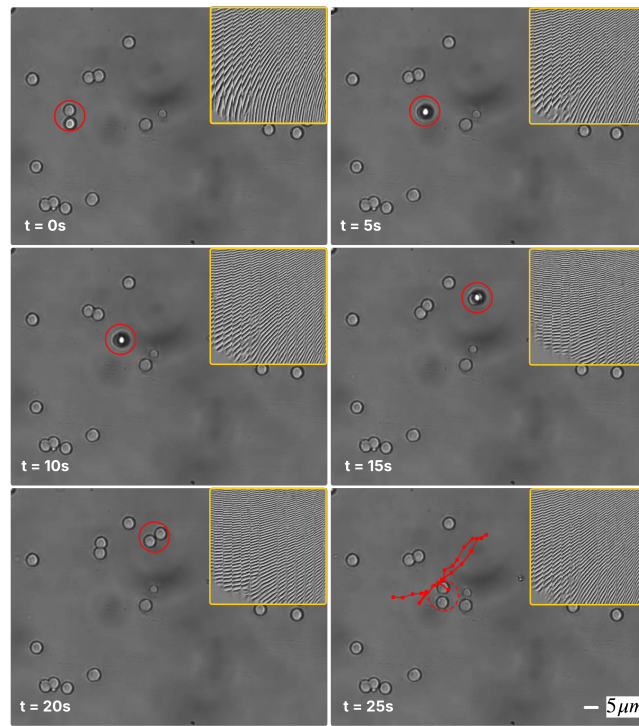


Figure 3.6: User controlled line trap trajectory time lapse. U-Net predictions used as the phase masks displayed onto the SLM highlighted outlined in orange. Red circles highlight the trapped beads. The overall trajectory is displayed at $t = 25\text{s}$ in red.

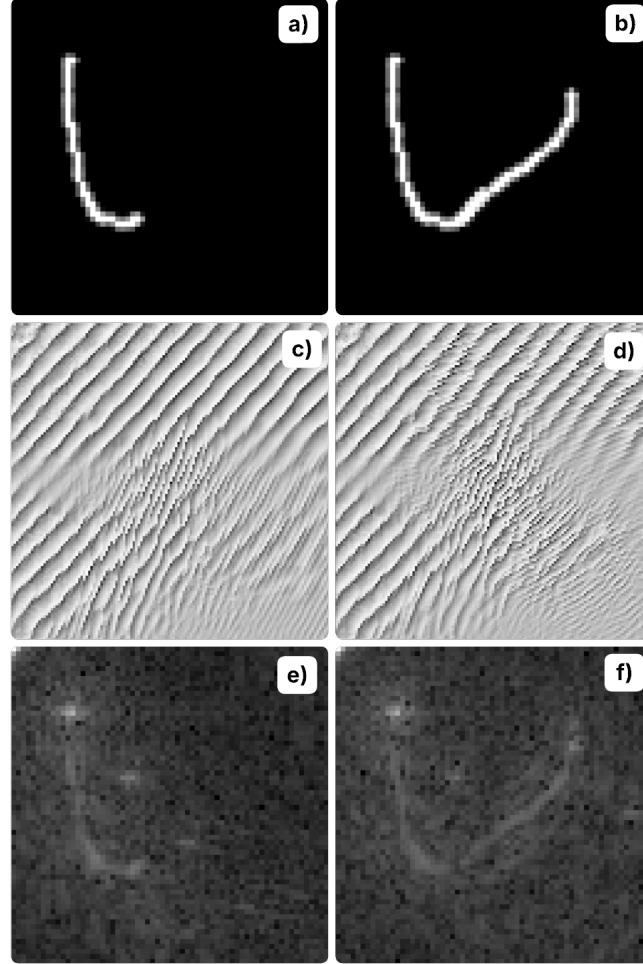


Figure 3.7: Phase mask predictions for complex intensity profiles using $D = 128$ model. a) Arbitrarily drawn intensity distribution. c) U-Net output given a) as input. e) Far-field simulation of c). b) Extension of a) where another spline is added arbitrarily. d) U-Net output given b) as input. f) Far-field simulation of d)

3.4 Conclusions

In summary, we provide the first-of-its-kind demonstration of phase mask prediction for a physical HOT system, using a carefully-structured encoder-decoder neural network. Our network incorporates the Zernike coefficients for aberration correction to accurately replicate the desired phase masks for different optical trap configurations. This work, therefore, opens a promising pathway to alleviate the curse of dimensionality in computing real-time holographic phase masks for miniaturized optical systems. In the future, it should be feasible to scale the model to 512×512 to match our HOT system’s SLM, given the advances in compute hardware and availability of strategies, such as progressive upscaling, to accelerate the training process. Training the U-Net using the GS algorithm’s phase outputs as the reference could also help the model generalize better to arbitrary intensity distributions, bringing the model closer to matching the capability of the GS algorithm in a single step. Last, it would be useful to integrate a real-time motion control algorithm, such as the model predictive controller in [2], for automated collision-free manipulation of a large number of microbeads using varying U-Net generated trap configurations.

Chapter 4

CONCLUSIONS

4.1 *Review of Contributions*

The core contributions of this work in the HOT field are as follows:

1. *Assistive automation tool and control framework for assembly of micro-structures using annular, line, and point traps.* Chapter 2 describes the strategy for enabling partial automation in the use of HOT systems for applications involving assembling formations of micro-objects. A software framework was developed to integrate the various components of the HOT system via a computer. The framework can be extended to additional hardware platforms for use on other HOT systems which may have different SLMs or optical configurations. A scalable iLQR enabled MPC obstacle avoidance and path planner enables rapid computation of solutions for multiple guided optical traps simultaneously through GPU acceleration.
2. *Novel U-Net based phase prediction model which reduces computation time of phase masks compared to existing algorithms that can also be used for more arbitrary intensity distributions.* Compared to existing baselines used for HOT systems, the GS and Red Tweezers algorithms, the U-Net exhibits a constant phase calculation time which is independent of the complexity or number of traps desired. In addition, we present a learning pipeline for generating reference and calibrated phase masks then simulating them to obtain intensity distributions analogous to those encountered during inference.

4.2 *Anticipated Impact*

HOTs are a prime system of interest for applications of controls due to the complex nature of the micro-environment which is constantly evolving to new states from Brownian motion

and external disturbances such as the incident laser beam. This complex nature makes manual human operation of the tweezers difficult and cumbersome, especially when dealing with workspaces containing large quantities of micro-objects. We anticipate this work will help alleviate the high human experience bar for reliable operation of HOTs and enable future experiments that would have been inhibited by complex workspaces.

Treatment of HOT systems in past works often directly take into account the stochasticity of the workspace by modeling freely diffusing particles with probability distributions or partially observable markov decision processes. However, this work takes a different approach to dealing with the uncertainty of the workspace by rapidly recomputing a controlled trap’s trajectory during each discrete movement. Therefore, the random Brownian motion of obstacle particles can be considered simply and greater control can be given to tuning the behavior of the optically guided object. Strategies leveraging GPU compute for quickly computing control actions for a system are a current area of research, especially for robotics and autonomous vehicle communities.

In Chapter 3, a methodology for training a U-Net to predict phase masks for HOTs is proposed and demonstrated. Deep learning models such as the U-Net applied to HOT phase retrieval is another area of research enabled by parallelized GPU compute. Our strategy enables rapid phase predictions with no performance impacts due to additional traps or added complexity. We anticipate that our work will pave the foundation for more powerful computation-enabled strategies to achieve more general autonomy in the HOT field.

4.3 Future Work

This work has demonstrated that assistive automation of HOTs can be achieved using multiple trap types with constant fluctuations in the positions of objects and the number of objects in view of the workspace camera. Setup of experiments were conducted manually by a human then passed to the control system to complete the desired operation.

While moving annular or line traps consisting of polystyrene micro-beads, there is often a phenomenon called optical binding which occurs due to off-resonant light impacting multiple micro-particle [13]. This optical binding can cause micro-particles to bind together at an undesirable location relative to the optical trap. For example, in a two-particle line trap, this

can cause one particle to be bound off-axis in multiple dimensions, requiring replacement of the trap. A future direction of the U-Net prediction model could be aimed at mitigating the effect of optical binding through modifying the light intensity of a line trap to achieve higher resistance to optical binding. In addition, structuring the phase output to the SLM with a direct input via desired intensity distributions at the far-field plane could easily be integrated to directly interface with other neural network-based control systems and would be a reasonable step towards a general end-to-end control model for HOTs.

BIBLIOGRAPHY

- [1] Ashis Banerjee, Sagar Chowdhury, and Satyandra K. Gupta. Optical tweezers: Autonomous robots for the manipulation of biological cells. *IEEE Robot. Autom. Mag.*, 21(3):81–88, 2014.
- [2] Ashis G. Banerjee, Keshav Rajasekaran, and Behnoosh Parsa. A step toward learning to control tens of optically actuated microrobots in three dimensions. In *IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, pages 1460–1465, 2018.
- [3] Ashis Gopal Banerjee, Sagar Chowdhury, Wolfgang Losert, and Satyandra K. Gupta. Real-time path planning for coordinated transport of multiple particles using optical tweezers. *IEEE Trans. Autom. Sci. Eng.*, 9(4):669–678, 2012.
- [4] Ashis Gopal Banerjee, Andrew Pomerance, Wolfgang Losert, and Satyandra K. Gupta. Developing a stochastic dynamic programming framework for optical tweezer-based automated particle transport operations. *IEEE Trans. Autom. Sci. Eng.*, 7(2):218–227, 2010.
- [5] Anna Bezryadina, John Keith, Joseph Chen, and Zhigang Chen. Optical manipulation of rod-shaped bacteria and adhesive cellular clusters with novel “tug-of-war” optical tweezers. In *Conf. Lasers Electro-Optics (CLEO) - Laser Sci. Photonic Applications*, pages 1–2, 2014.
- [6] Richard W. Bowman, Graham M. Gibson, Anna Linnenberger, David B. Phillips, James A. Grieve, David M. Carberry, Steven Serati, Mervyn J. Miles, and Miles J. Padgett. “Red tweezers”: Fast, customisable hologram generation for optical tweezers. *Comput. Phys. Commun.*, 185(1):268–273, 2014.
- [7] Haoyao Chen, Can Wang, Xiaojian Li, and Dong Sun. Transportation of multiple biological cells through saturation-controlled optical tweezers in crowded microenvironments. *IEEE/ASME Trans. Mechatronics*, 21(2):888–899, 2016.
- [8] Wen Chen. 3d gerchberg-saxton optical correlation. *IEEE Photonics Journal*, 10(2):1–9, 2018.
- [9] Samira Chizari, Miles P. Lim, Lucas A. Shaw, Sydney P. Austin, and Jonathan B. Hopkins. Automated optical-tweezers assembly of engineered microgranular crystals. *Small*, 16(25):2000314, 2020.

- [10] Jennifer E. Curtis and David G. Grier. Structure of optical vortices. *Phys. Rev. Lett.*, 90:133901, Apr 2003.
- [11] David Eberly. Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid, 2020.
- [12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.
- [13] Kayn A. Forbes, David S. Bradshaw, and David L. Andrews. Optical binding of nanoparticles. *Nanophotonics*, 9(1):1–17, 2020.
- [14] R. W. Gerchberg and W. O. Saxton. A practical algorithm for the determination of phase from image and diffraction plane pictures. *Optik*, 35(2):237–246, 1972. Received 29 November 1971.
- [15] David G Grier. A revolution in optical manipulation. *Nature*, 424(6950):810–816, 2003.
- [16] Cheng-Shan Guo, Ya-Nan Yu, and Zhengping Hong. Optical sorting using an array of optical vortices with fractional topological charge. *Opt. Commun.*, 283(9):1889–1893, 2010.
- [17] Songyu Hu, Shuxun Chen, Si Chen, Gang Xu, and Dong Sun. Automated transportation of multiple cell types using a robot-aided cell manipulation system with holographic optical tweezers. *IEEE/ASME Trans. Mechatronics*, 22(2):804–814, 2017.
- [18] Tao Ju, Shuang Liu, Jie Yang, and Dong Sun. Rapidly exploring random tree algorithm-based path planning for robot-aided optical manipulation of biological cells. *IEEE Trans. Autom. Sci. Eng.*, 11(3):649–657, 2014.
- [19] Kosta Ladavac and David G. Grier. Microoptomechanical pumps assembled and driven by holographic optical vortex arrays. *Opt. Express*, 12(6):1144–1149, Mar 2004.
- [20] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
- [21] Jonathan Leach, Kurt Wulff, Gavin Sinclair, Pamela Jordan, Johannes Courtial, Laura Thomson, Graham Gibson, Kayode Karunwi, Jon Cooper, Zsolt John Laczik, and Miles Padgett. Interactive approach to optical tweezers control. *Appl. Opt.*, 45(5):897–903, Feb 2006.
- [22] Isaac C.D. Lenton, Alexander B. Stilgoe, Timo A. Nieminen, and Halina Rubinsztein-Dunlop. OTSLM toolbox for structured light methods. *Comput. Phys. Commun.*, 253:107199, 2020.

- [23] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for non-linear biological movement systems. In *First International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004.
- [24] Xiang Li and Chien Chern Cheah. A simple trapping and manipulation method of biological cell using robot-assisted optical tweezers: Singular perturbation approach. *IEEE Trans. Ind. Electron.*, 64(2):1656–1663, 2017.
- [25] Xiaojian Li and Dong Sun. Automated in-vivo transportation control of biological cells using robot-aided optical tweezers. In *Autonomous Robot-Aided Optical Manipulation for Biological Cells*, pages 93–113. 2021.
- [26] Jeffrey Melzer and Euan Mcleod. Assembly of multicomponent structures from hundreds of micron-scale building blocks using optical tweezers. *Microsyst. Nanoeng.*, 7:45, 06 2021.
- [27] Giuseppe Pesce, Philip H Jones, Onofrio Maria Maragò, and Giovanni Volpe. Optical tweezers: Theory and practice. *Eur. Phys. J. Plus*, 135(949), 2020.
- [28] Keshav Rajasekaran, Ekta Samani, Manasa Bollavaram, John Stewart, and Ashis G. Banerjee. An accurate perception method for low contrast bright field microscopy in heterogeneous microenvironments. *Appl. Sci.*, 7(12), 2017.
- [29] Keshav Rajasekaran, Ekta Umesh Samani, John Stewart, and Ashis G. Banerjee. Imaging-guided collision-free transport of multiple optically trapped beads. In *Int. Conf. Manipulation Autom. Robot. Small Scales (MARSS)*, pages 1–6, 2017.
- [30] Sebastien Rancourt-Grenier, Ming-Tzo Wei, Jar-Jin Bai, Arthur Chiou, Paul P. Bareil, Pierre-Luc Duval, and Yunlong Sheng. Dynamic deformation of red blood cell in dual-trap optical tweezers. *Opt. Express*, 18(10):10462–10472, May 2010.
- [31] Yunxiao Ren, Meysam Keshavarz, Salzitsa Anastasova, Ghazal Hatami, Benny Lo, and Dandan Zhang. Machine learning-based real-time localization and automatic trapping of multiple microrobots in optical tweezer. In *Int. Conf. Manipulation Autom. Robot. Small Scales (MARSS)*, pages 1–6, 2022.
- [32] Ana Karen Reyes, Carmen E. Domínguez-Flores, Juan A. Rayas, David Monzón-Hernández, Amalia Martínez-García, and Raúl R. Cordero. Real-time temperature monitoring in an optical trap. *IEEE Photon. Technol. Lett.*, 34(2):121–124, 2022.
- [33] Yohai Roichman and David G Grier. Projecting extended optical traps with shape-phase holography. *Opt. Lett.*, 31(11):1675–1677, 2006.
- [34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

- [35] Edward Schmerling. Heuristic-based nonlinear mpc for "asteroids".ipynb. <https://github.com/StanfordASL/AA203-Examples>, 2023.
- [36] Hullas Sehgal. *Optical tweezers: Characterization and systems approach to high bandwidth force estimation*. PhD thesis, University of Minnesota, 2010.
- [37] Yinan Shen, David A. Weitz, Nancy R. Forde, and Marjan Shayegan. Line optical tweezers as controllable micromachines: techniques and emerging trends. *Soft Matter*, 18:5359–5365, 2022.
- [38] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913, 2012.
- [39] Xiaolin Wang, Zuankai Wang, and Dong Sun. Cell sorting with combined optical tweezers and microfluidic chip technologies. In *11th Int. Conf. Control Autom. Robot. Vis.*, pages 201–206, 2010.
- [40] Yanhua Wu, Dong Sun, Wenhao Huang, and Youfu Li. Path planning in automated manipulation of biological cells with optical tweezers. In *IEEE Int. Conf. Control Autom.*, pages 2021–2026, 2009.
- [41] Mingyang Xie, James K. Mills, Yong Wang, Masih Mahmoodi, and Dong Sun. Automated translational and rotational control of biological cells with a robot-aided optical tweezers manipulation system. *IEEE Trans. Autom. Sci. Eng.*, 13(2):543–551, 2016.
- [42] Mingyang Xie, Adnan Shakoor, and Changcheng Wu. Manipulation of biological cells using a robot-aided optical tweezers system. *Micromachines*, 9(5), 2018.
- [43] Mingyang Xie, Yong Wang, Gang Feng, and Dong Sun. Automated pairing manipulation of biological cells with a robot-tweezers manipulation system. *IEEE/ASME Trans. Mechatronics*, 20(5):2242–2251, 2015.
- [44] Tommy Zhang, Nicole Werner, and Ashis G. Banerjee. Toward automated formation of composite micro-structures using holographic optical tweezers. In *2024 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS)*, pages 1–6, 2024.
- [45] Tommy Zhang, Nicole Werner, and Ashis G. Banerjee. Toward automated formation of composite micro-structures using holographic optical tweezers. In *Int. Conf. Manipulation Autom. Robot. Small Scales (MARSS)*, pages 1–6, 2024.