University of Southampton

Faculty of Engineering and Physical Sciences

Optoelectronics Research Centre

Image-based Neural Networks for Monitoring and Controlling Light-matter Interactions

by

Yunhui Xie

ORCID ID 0000-0002-8841-7235

Thesis for the degree of Doctor of Philosophy

March 2022

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences

Optoelectronics Research Centre

Doctor of Philosophy

Image-based Neural Networks for Monitoring and Controlling Light-matter Interactions

by

Yunhui Xie

Recent advances in the field of deep learning have unlocked an abundance of exciting novel scientific applications. The convolutional neural network (CNN), which is a type of neural network optimised for processing two-dimensional data such as images, has been at the forefront of many of these developments. A fundamental capability of the CNN is the ability to extract numerical descriptions directly from real-world observations, which, as shown in this thesis, enables a wide range of techniques based on the real-time measurement, optimisation, and process control of experiments. Critically, the CNN is trained on data (experimental or theoretical), and hence can be applied in cases where the complexity of the experiment may prevent modelling approaches based on a fundamental understanding of the system. This thesis provides four experimental applications of CNNs across the fields of laser machining, laser-based sensing, and laser-based manipulation, for both supervised learning and reinforcement learning.

Firstly, that a CNN can accurately identify the type and concentration of microparticles in a solution, directly from processing the scattering pattern when the solution is illuminated by laser light. Secondly, that a CNN can identify the translation and rotation of a laser beam profile during laser machining, and also automatically cease laser machining when tasked with machining through the top layer of a multilayer sample of unknown thickness. Thirdly, that a CNN, via reinforcement learning, can learn how to translate a microparticle through a maze, whilst avoiding collisions with other microparticles, through the use of the optical tweezers effect. Finally, that a CNN, via reinforcement learning, can optimise the toolpath generation for laser machining, when tasked with manufacturing a specific target pattern, whilst also correcting for errors in real-time.

Table of Contents

Contents

Table of Tables

Table of Figures

Table of Figures

Research Thesis: Declaration of Authorship

Print name: Yunhui Xie

Title of thesis: Image-based Neural Networks for Monitoring and Controlling Light-matter Interaction

I declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:

    1) Grant-Jacob J, **Xie Y**, Mackay B et al. Particle and salinity sensing for the marine environment via deep learning using a Raspberry Pi. Environmental Research Communications. 2019;1(3):035001. Doi: 10.1088/2515-7620/ab14c9
    2) **Xie Y**, Heath D, Grant-Jacob J et al. Deep learning for the monitoring and process control of femtosecond laser machining. Journal of Physics: Photonics. 2019;1(3):035002. Doi: 10.1088/2515-7647/ab281a
    3) **Xie Y**, Heath D, Grant-Jacob J, et al. Laser Processing using Machine Learning for Real-Time Monitoring and Control. 6TH UK Industrial Laser Symposium (ILAS) 2019. https://ilas2019.co.uk/wp/wp-content/uploads/2019/01/Xie-Yunhui.pdf
    4) Praeger M*, **Xie Y***, Grant-Jacob J, Eason R, Mills B. Playing optical tweezers with deep reinforcement learning: in virtual, physical and augmented environments. Machine Learning: Science and Technology. 2021;2(3):035024. Doi:10.1088/2632-2153/abf0f6
    5) **Xie Y**, Praeger M, Grant-Jacob J, Eason R, Mills B. Motion control for laser machining via reinforcement learning. Optics Express. 2022;30(12):20963. Doi: 10.1364/OE.454793

6) **Xie Y**, Praeger M, Grant-Jacob J, et al. Autonomous and Self-correcting Laser Subtractive Patterning Using Reinforcement Learning. Conference for Lasers and Electro-Optics (CLEO) 2022. https://ieeexplore.ieee.org/document/9890972

Signature: .....Date:.....

Definitions and Abbreviations

ω ..... Angular frequency

c ..... Speed of light in vacuum

L ..... Length of the optical cavity

V ..... Volt

n ..... Refractive index

n_0 ..... Linear refractive index

n_2 ..... Second-order nonlinearity of refractive index

I ..... Intensity of electromagnetic field

P ..... Power

E ..... Energy

$E(\cdot)$ ..... Electric field

$E_{x,y,z}$ ..... Electric field along x, y, or z direction

$A_{\parallel}, A_{\perp}$ ..... Amplitude of p and s polarised incident electric filed

$R_{\parallel}, R_{\perp}$ ..... Amplitude of p and s polarised reflected electric filed

E^i, E^r ..... Incident and reflected electric field

τ ..... Pulse duration

τ^i, τ^r ..... Phase components of incident and reflected electric field

t ..... Time

k ..... Wavenumber

$\hbar$ ..... Reduced Planck constant

w_0 ..... Beam waist

δ ..... Optical path difference

f ..... Focal length

R ..... Radius of curvature

$N.A$ ..... Numerical aperture

J ..... Joule

mJ ..... Millijoule

W ..... Watt

mW ..... Milliwatt

s ..... Second

ms ..... Millisecond

ps ..... Picosecond

fs ..... Femtosecond

m ..... Metre

cm ..... Centimetre

mm ..... Millimetre

μm ..... Micrometre

nm ..... Nanometre

$\mathbb{R}$ ..... Real number set

$\mathbb{Z}$ ..... Integer number set

$\mathbb{Z}^+$ ..... Positive integer set

$\mathcal{N}(\mu, \sigma)$ ..... Normal distribution, mean of μ and variance of σ^2

$\mathcal{U}_{[a,b)}$ ..... Continuous uniform distribution in the interval $[a, b)$

Chapter 1 Introduction

1.1 Motivation

Light Amplification by Stimulated Emission of Radiation is a physical process where excited atoms or ions emit photons through the stimulated emission (Svelto and Hanna, 1998). The radiated photons possess some unique properties that are of great scientific and manufacturing importance. Generally speaking, the laser is commonly characterised as electromagnetic radiation that bears excellent spatial coherence and possesses high photon number per cavity mode (Svelto and Hanna, 1998). These characteristics enable laser light to propagate through space (or mediums) over a substantial distance in a confined volume, and to deliver concentrated energy. Therefore, there have been diverse applications based on laser light, for instance, laser sensing (Melle et al., 1993, Jono et al., 2006, Cranch et al., 2008), laser materials processing (Sun and Ion, 1995, Sugioka and Cheng, 2012, Žemaitis et al., 2019) and laser communications (McElroy et al., 1977, Jono et al., 2006). Some applications in relation to laser light involve moveable experimental apparatuses for positioning the beam focus (Žemaitis et al., 2019), for controlling beam shape (Weiner, 2000) and wavefront (Yatooshi et al., 2015) and for aligning beam-delivering optics (Morrison et al., 1994) etc. The manoeuvre of these moveable apparatuses usually relies heavily upon experienced manual labour in terms of their tool-path designs and/or controls. In addition, in some cases where the experiment apparatus follows a predetermined sequence of actions, the occurrences of hardware and/or software malfunctions usually needs to be assessed and addressed with extra caution. Otherwise, any incorrectly executed actions that stem from poorly controlled experimental apparatus could result in flawed or even defective final products. Studies have been conducted in a broad field of areas in optics to ease the burden of reliance on experienced manual labour that is needed for supervising optical processes — the aim of these works usually revolve around devising a monitoring system that closely observes an optical system (through cameras or sensors) and building a feedback loop upon this monitoring system that maintains the optical system in a steady state (Song and Mazumder, 2010, You et al., 2014, Reutzel and Nassar, 2015).

In recent years, studies of artificial intelligence (AI) have gained popularity. Amongst the wide spectrum of AI studies, neural network (NN) and deep neural network (DNN) are arguably the most successful and popular AI architecture. Loosely inspired by neuroscience, a neural network is a data-driven computing paradigm that possesses a set of interconnected nonlinear functions, each of which contains tuneable parameters (Goodfellow et al., 2016). This enormous parameter space offers an incredible capability for a neural network to approximate other functions. This universal approximation ability can thus be utilised to construct computing models of great complexity;

recent breakthroughs in many research fields such as computer vision (Krizhevsky et al., 2012, Szegedy et al., 2015, He et al., 2016), natural language processing (Radford et al., 2019, Brown et al., 2020), autonomous mobility (Hadsell et al., 2009, Farabet et al., 2012, Bojarski et al., 2016), and content filtering (Chau and Chen, 2008, Bansal et al., 2016) leverage the function approximating capabilities of deep neural networks to extract and represent abstractive information from raw data that only human can previously fully understand (e.g., human languages) (LeCun et al., 2015). To achieve these similar functionalities with conventional methods would sometimes be very challenging, in the sense that it could be a mountainous task to manually compile the associated programmatical descriptions of the underlying processes.

A major milestone in the studies of artificial intelligence is human-competitive performance in perfect (Silver et al., 2018) and imperfect information games (Berner et al., 2019, Vinyals et al., 2019) that are achieved by Reinforcement Learning (RL). In 2016, a Go agent that was developed by DeepMind, Alpha Go (Silver et al., 2017), won a five-match game in the game of Go against Lee Sedol who had won 18 world championships by that time. Reinforcement Learning is an area of research that focuses on the automated search for a set of strategies (i.e., an adaptive policy) that achieve certain goals in an external environment (Sutton and Barto, 2018). A typical reinforcement learning agent interacts with an outside environment through a recurring cycle which is composed of the following steps — observing the environment, deducing an appropriate action in accordance with the observation(s) of the environment and exerting the deduced action to the environment. This scheme of decision-making conceptually coincides with other decision-making models, most notably the Observe–Orient–Decide–Act Loop (OODA Loop) that was initially developed by United States Air Force (Osinga, 2007), but was later widely adopted in other areas (e.g., business, litigation). The innovative concept of the reinforcement learning (i.e., asynchronous online dynamic programming (Christopher, 1992, Pendrith and Ryan, 1997)), however, is that it utilises a figure of merit, which is commonly referred to as a ‘reward’, to mathematically describe how well a predefined goal has been achieved by an exerted action. And thus, the pursuit of a pre-defined goal can be approximated by the pursuit of a higher amount of reward, which transforms a decision-making problem into a concave function optimisation problem. A policy in a reinforcement learning agent can therefore be mathematically described as a function that is able to map an input observation of an environment to an output action that leads to the maximum amount of reward which the reinforcement Learning agent is able to obtain in the future.

Machine learning has been used extensively across almost all research fields over the past decades. In recent years, deep learning has been increasingly applied to scientific research, resulting in a rapidly increasing number of publications and studies. The field of photonics is no exception, and there are now many international research groups focussing on the application of

deep learning for solving complex photonics challenges. Accordingly, there have been many comprehensive deep learning review articles, see for example ultrafast lasers (Genty et al., 2021), laser machining (Mills and Grant‐Jacob, 2021), photonic design (Ma et al., 2021), nanophotonics (Yao et al., 2019), neuromorphic computing (Shastri et al., 2021), biomedicine (Mackay et al., 2021) and microscopy (Xing et al., 2017).

Of course, the work in this thesis strongly complements the activity of the research group of the author. The research group at the led by Dr Ben Mills at the University of Southampton has pioneered many of the key deep learning results in the fields of femtosecond laser machining (Heath et al., 2018, Mills et al., 2018a, Mills et al., 2018b, Xie et al., 2019, McDonnell et al., 2021b, McDonnell et al., 2020, Mills et al., 2022, McDonnell et al., 2021a), whilst also contributing strongly to the fields of sensing (Grant-Jacob et al., 2018, Grant-Jacob et al., 2019c, Grant-Jacob et al., 2019a, Grant-Jacob et al., 2020, Grant-Jacob et al., 2021a, Grant-Jacob et al., 2021b), microscopy (Grant-Jacob et al., 2019b, Praeger et al., 2021, Buchnev et al., 2022), and biomedicine (Mackay et al., 2020b, Mackay et al., 2020a). Indeed, the work presented in this thesis builds upon much of the existing work from this research group.

1.2 Thesis Outline

This thesis is written as follows. Chapter 2 introduces the key components of the interaction of light and matter. Chapter 3 provides a discussion of the relevant machine learning and deep learning approaches used throughout the experimental chapters. Chapters 4, 5, 6 and 7 represent separate experimental results in the application of deep learning for a range of laser-based techniques. Specifically, Chapter 4 shows the application of a convolutional neural network for sensing of particulates in a liquid via processing of the scattered light. Chapter 5 shows the use of a convolutional neural network for a range of techniques for compensation of errors and demonstration of techniques for laser beam shaping when used for laser machining. Chapter 6 presents the application of reinforcement learning for the control of microparticles via the optical tweezers effect. Chapter 7 presents the application of reinforcement learning for the spatial control of laser machining for the machining of bespoke shapes. The separation of work into these chapters is intentional, as chapters 4 and 5 correspond to the use of convolutional neural networks and chapters 6 and 7 correspond to the use of reinforcement learning. In addition, chapters 4 and 6 correspond to the non-destructive interaction of light and matter, whilst chapters 5 and 7 correspond to the destructive interaction (i.e., laser machining). Whilst each experimental chapter contains a specific conclusion, Chapter 8 provides a general conclusion and a discussion of future work.

Chapter 1

Published work associated with each experimental chapter is as follows.

Chapter 4: 'Microparticle Sensing in Solution via Convolutional Neural Network'

1) Grant-Jacob J, **Xie Y**, Mackay B et al. Particle and salinity sensing for the marine environment via deep learning using a Raspberry Pi. Environmental Research Communications. 2019;1(3):035001. Doi: 10.1088/2515-7620/ab14c9

Chapter 5: 'Convolutional Neural Networks for Monitoring and Controlling Laser Machining'

2) **Xie Y**, Heath D, Grant-Jacob J et al. Deep learning for the monitoring and process control of femtosecond laser machining. Journal of Physics: Photonics. 2019;1(3):035002. Doi: 10.1088/2515-7647/ab281a
3) **Xie Y**, Heath D, Grant-Jacob J, et al. Laser Processing using Machine Learning for Real-Time Monitoring and Control. 6TH UK Industrial Laser Symposium (ILAS) 2019. https://ilas2019.co.uk/wp/wp-content/uploads/2019/01/Xie-Yunhui.pdf

Chapter 6: 'Controlling Optical Tweezers using Reinforcement Learning'

4) Praeger M*, **Xie Y***, Grant-Jacob J, Eason R, Mills B. Playing optical tweezers with deep reinforcement learning: in virtual, physical and augmented environments. Machine Learning: Science and Technology. 2021;2(3):035024. Doi:10.1088/2632-2153/abf0f6 (*****equal contribution and joint first-name authorship)

Chapter 7: 'Tool-Path Generation for Femtosecond Laser Machining'

5) **Xie Y**, Praeger M, Grant-Jacob J, Eason R, Mills B. Motion control for laser machining via reinforcement learning. Optics Express. 2022;30(12):20963. Doi: 10.1364/OE.454793
6) **Xie Y**, Praeger M, Grant-Jacob J, et al. Autonomous and Self-correcting Laser Subtractive Patterning Using Reinforcement Learning. Conference for Lasers and Electro-Optics (CLEO) 2022. https://ieeexplore.ieee.org/document/9890972

Chapter 2 Interaction of Light and Matter

This chapter provides a technical introduction to the interaction of light and matter relevant to the experimental results presented in this thesis. Of course, the topic of the interaction of light and matter is extraordinarily broad, and as such there are a wealth of textbooks written in this area. The purpose of this chapter is therefore to provide a specific set of knowledge for the reader, to help understand the more complex components of the experimental chapters. This chapter begins with a discussion of lasers (including femtosecond lasers and diode lasers) before moving onto beam shaping (with a particular emphasis on the application of a digital micromirror device for spatial intensity profile shaping). Following this, introductions to Mie scattering (relevant for Chapter 4), optical tweezers (relevant for Chapter 6), and laser machining (relevant for Chapters 5 and 7) are provided. Finally, a conclusion is presented.

2.1 Lasers

In general, the simplest optical cavity consists of two parallel mirrors that are positioned in front of each other along an optical axis. This arrangement of mirrors allows waves to propagate back and forth between the mirrors, and allows waves with certain properties to be iteratively amplified in order to produce an intense output (Svelto and Hanna, 1998). In this two-mirror cavity, which is assumed to be stable (i.e., it employs ideal mirrors), along the axial direction, the permitted longitudinal cavity modes (see Appendix A1 for proof) have angular frequencies of:

$$\omega = \frac{\pi c}{L} \cdot q \qquad (2-1)$$

where c is the velocity of light and L is the length of the cavity (Lamb Jr, 1964). If ω_0 is denoted as the angular frequency of the first cavity mode where $q = 1$, the angular frequency of the n^{th} cavity mode can be therefore described by $\omega_n = \omega_0 + n\Delta\omega$ where $\Delta\omega = {}^{\pi c}/_{L}$. The temporal profiles of the first 5 cavity modes with the angular frequencies of ω_0, ω_1, ω_2, ω_3 and ω_4 respectively are illustrated in Figure 2-1 (assuming a unity amplitude of electric field and a unity cavity length L).

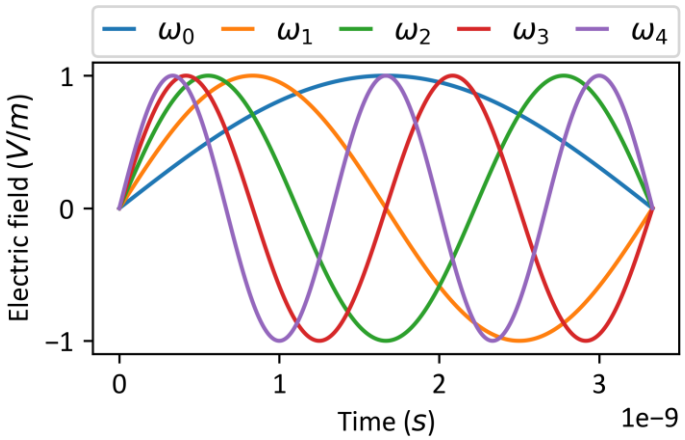

Figure 2-1 Electric fields of the longitudinal cavity modes whose angular frequencies are $\pi c/L$, $2\pi c/L$, $3\pi c/L$, $4\pi c/L$ and $5\pi c/L$ for ω_0, ω_1, ω_2, ω_3 and ω_4 respectively.

Assuming that $E(\omega_0)$, $E(\omega_1)$, $E(\omega_2)$, $E(\omega_3)$ and $E(\omega_4)$ are all oscillating in-phase, the constructive interferences of these 5 cavity modes yields a strong electric pulse that circulates inside the cavity. Figure 2-2 a) depicts the super-position of the electric fields of these cavity modes (i.e., $E(\omega_0) + E(\omega_1) + E(\omega_2) + E(\omega_3) + E(\omega_4)$). Comparatively, $\mathrm{E}(\omega_q, \Delta\omega, N)$ gives a mathematical description of the resultant circulating electric pulse (see Appendix A2 for derivation), which are produced by the constructive interferences of the N consecutive cavity modes, starting from $\mathrm{E}(\omega_q)$ and spaced by $\Delta\omega$. Figure 2-2 b) shows $\mathrm{E}(\omega_q = \omega_0, \Delta\omega = \pi c/L, N = 5)$ which is equivalent to $E(\omega_0) + E(\omega_1) + E(\omega_2) + E(\omega_3) + E(\omega_4)$.

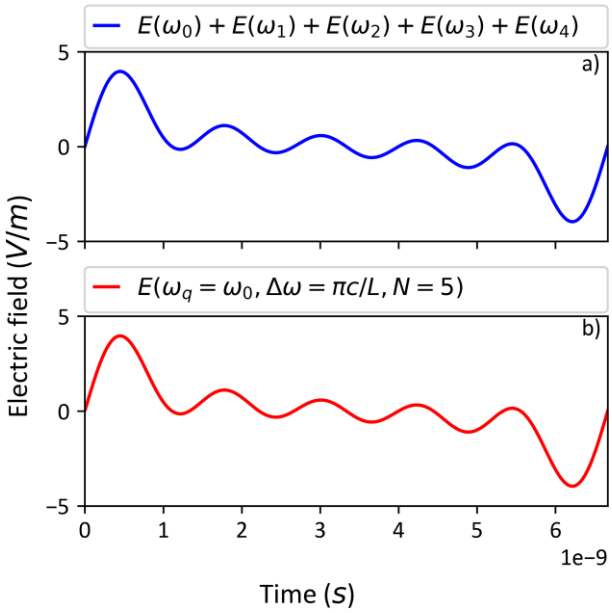

Figure 2-2 Constructive interfrences between cavity modes oscillating in-phase produces a strong pulse in the temporal dimension, showing for a) $E(\omega_0) + E(\omega_1) + E(\omega_2) + E(\omega_3) + E(\omega_4)$ and b) $\mathrm{E}(\omega_q = \omega_0, \Delta\omega = \pi c/L, N = 5)$.

From equation A 8 (see Appendix A2), if $\omega_q \gg {}^{(N-1)\Delta\omega}/_{2}$, the mathematical expression of the circulating pulse can be approximated by (Akhmanov et al., 1992):

$$\mathrm{E}(\omega_q, \Delta\omega, N) \sim \exp\left[-i\left(\omega_q t - \frac{\pi}{2}\right)\right]\left(\frac{\sin({}^{N\Delta\omega t}/_{2})}{\sin({}^{\Delta\omega t}/_{2})}\right) \qquad (2-2)$$

Note that $\omega_q + \frac{(N-1)\Delta\omega}{2}$ is effectively the angular frequency of the electric field that possesses the centre wavelength, but in order to be consistent with the previously defined mathematical expression $\mathrm{E}(\omega_q, \Delta\omega, N)$ the centre wavelength is substituted by the angular frequency of the first electric field ω_q as an approximation. The modifying term $\frac{\sin({}^{N\Delta\omega t}/_{2})}{\sin({}^{\Delta\omega t}/_{2})}$ in equation 2-2 (i.e., the complex amplitude) can be proved to have $\left|\frac{\sin({}^{N\Delta\omega t}/_{2})}{\sin({}^{\Delta\omega t}/_{2})}\right| \leq N$ (see Figure 2-3). That is, the more modes there are oscillating in-phase, the higher the maximum amplitude of the electric field of the resultant pulse. In addition, the duration of the pulse can be estimated via evaluating the horizontal distance between the highest and lowest points of the pulse on the time domain. That is, $\left|\sin({}^{N\Delta\omega t}/_{2})\right| = 0$ subject to $\left|\sin({}^{\Delta\omega t}/_{2})\right| \neq 0$, which gives half width of the duration of the pulse to be $\frac{2\pi}{N\Delta\omega}$ (see Figure 2-3). That is, the duration of the pulse is inversely proportional to the number of modes that are oscillating in-phase. In conclusion, a series of cavity mode that are oscillating in-phase enables the production of a train of short-duration, high peak power laser pulses.

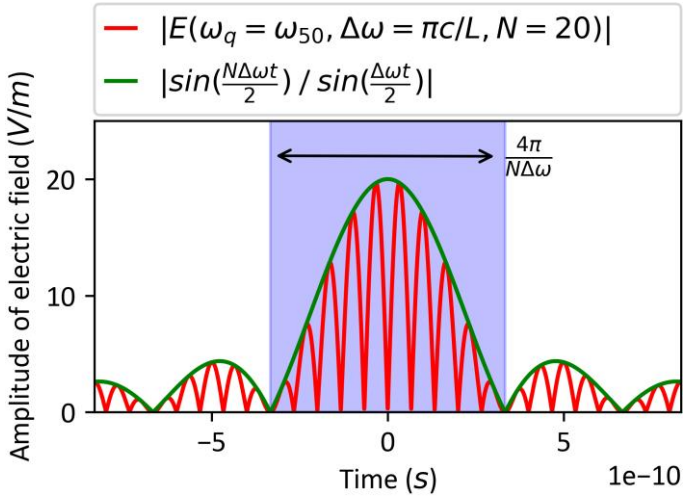

Figure 2-3 The pulsed electric field, which is discribed by $\left|\mathrm{E}(\omega_q = \omega_0, \Delta\omega = {}^{\pi c}/_{L}, N = 5)\right|$, is colour-coded in red, the maginitude of the modifying term $\left|\frac{\sin({}^{N\Delta\omega t}/_{2})}{\sin({}^{\Delta\omega t}/_{2})}\right|$ (i.e., the complex amplitude) is colour-coded in green and the termporal span $\frac{4\pi}{N\Delta\omega}$ of the pulse (the duration of the pulse) in shaded in blue.

By this method, the formation of the laser pulses is therefore requires modulating the phase relationships between cavity modes, and hence the name 'mode-locking' is given to such techniques that lead to the generation of optical pulses in a laser cavity (Lamb Jr, 1964). A class of the 'mode-locking' techniques that are able to produce a train of pico- to femto- second laser pulses is 'passive mode-locking' where a saturable absorber is usually placed in the laser cavity (Ippen et al., 1972). The attenuation of a pulse by the saturable absorber is a function of the optical power of the pulse — a pulse with a low optical power experiences a high absorption, which may result in a total loss that is higher than the gain for the pulse; on the contrary, a pulse with a high optical power is attenuated to a much less extent, and thus can be amplified if the total loss that the pulse experiences is lower than the gain. This scheme of pumping power distribution ensures that a pulse with a high optical power, and its composing cavity modes, can be iteratively amplified, resulting in a train of short-duration, high peak power laser pulses. Kerr-lens mode-locking is a method that utilises the Kerr effect to enable passive mode-locking (Salin et al., 1991). The Kerr effect describes an increase or a decrease to the refractive index of a medium when an intense electromagnetic field (with an intensity of I) propagates through the medium:

$$n = n_0 + n_2 I \qquad (2-3)$$

where n_0 and n_2 are the linear and second-order nonlinear components of the refractive index respectively. If n_2 is positive, a plate of this medium of uniform thickness, when illuminated by an intense Gaussian beam (i.e., TEM_{00}) in a normal incident, acts effectively as a converging lens that is able to focus the beam. This situation arises as the spatial centre of the beam experiences a higher refractive index than the spatial outer region of the beam. In such a case, the focal length of the Kerr lens is a function of the instantaneous power of the beam (i.e., self-focusing effect, see Appendix A3 for details). An aperture can therefore be placed in the cavity at the beam focus of the pulse associated with a high optical power, hence allowing this high-energy pulse to repeatedly propagate through without experiencing high losses. On the contrary, owing to the fact that the aperture is placed at a position where other pulses with lower optical power are not in focus, this aperture can therefore limit the transmissions of these pulses, hence resulting in higher total losses for these pulses.

In order to obtain a train of energetic and short-duration laser pulses, further amplification can be applied to the generated laser pulses. Due to the high peak power of an energetic and short-duration pulses (assuming a Gaussian temporal profile, the optical power of a pulses with a pulse energy E = 2 mJ and pulse duration τ = 150 fs is approximately 0.94 × E/τ ≈ 12.5 GW), the damage threshold of the apparatus inside an optical cavity (e.g., mirrors and crystals) can be easily exceeded (for example, the pumping fluence of a Ti:sapphire gain medium is usually ~1 J/cm^2 or 1 Ws^{-1}cm^{-2}

(Canova et al., 2006)). Therefore, additional measures are usually required in order to amplify these high peak power pulses, for example chirped pulse amplification (CPA). Chirp pulse amplification manipulates the duration of a pulse through stretching and compressing the pulse via a set of gratings or prisms (Strickland and Mourou, 1985). A pair of gratings or prisms can introduce optical path differences, and hence phase differences, to frequency components of a pulse, resulting in a chirped pulse or an unchirped pulse depending on the dispersive property of the set of gratings or prisms (i.e., stretching or compressing). A chirped pulse has a longer temporal profile compared to that of the original pulse before it is stretched, and if the temporal shape of the chirped pulse remains unchanged (e.g., Gaussian), the peak power of the chirped pulse is therefore reduced. This enables the chirped pulse to be safely amplified without damaging the amplifier apparatus. The short duration can be restored for the amplified, chirped pulses via a similar process where the opposite dispersion is introduced to these pulses by a set of gratings or prisms with the opposite dispersive property (see Figure 2-4). Novel applications of femtosecond laser pulses for laser machining are presented in Chapter 5 and Chapter 7.

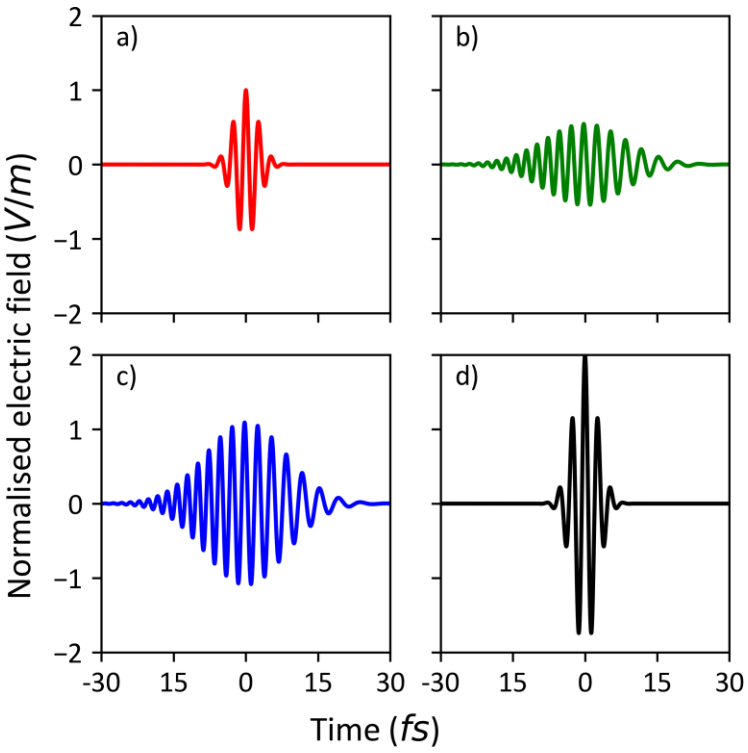

Figure 2-4 Simple depiction of the CPA prosess: a) A pulse of a Gaussian temporal profile with no chirp. b) A chirped pulse that is produced by introducing a positive dispersion to the pulse in a). c) Homogeneous amplification to the chirped pulse in b), which yields an amplified pulse whose maximum amplitude roughly equals to that of the pulse with no chirp in a). d) Applying negative disperison to the amplified, chirped pulse in c), which restores the short duration of the initial pulse in a) but the resultant pulse possesses a much larger amplitude.

2.2 Spatial Intensity Shaping

The Digital-Micromirror Devices (DMD) is a Micro-Electron Mechanical System (MEMS) device that is composed of a mosaic of mirrors (Yoder et al., 2001). Each micro-sized mirror is of few micrometres in length (see Figure 2-5). Each individual micromirror is mounted on a supporting post that is jointed on a torsion hinge. Applying a voltage to one of the address electrodes exerts electrostatic force to one corner of the micromirror, causing the micromirror to be rotated about the hinge axis. Application of a voltage to the opposite electrode reverses the process, hence allowing the micromirror to be flipped between the two positions on demand. The result of this motion is an angular offset between the normal of the tilted DMD mirror (i.e., facet normal) and the normal of the DMD array (i.e., surface normal). As all mirrors in a DMD can be independently operated, a DMD can therefore function as a programmable Spatial Light Modulator (SLM), if an appropriate experimental arrangement is applied.

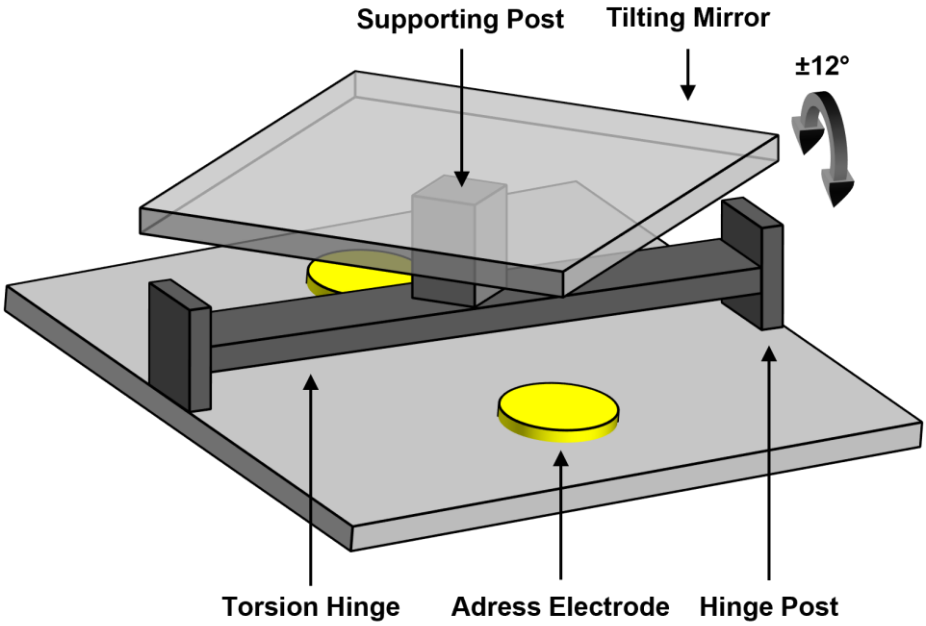

Figure 2-5 Schematics of a micro-sized mirrors on a DMD. The micro-szied mirror is mounted on a rotary torsion hinge via a supporting hinge, enabling the micro-sized mirrors to be rotated about the hinge axis. A pair of address electrodes are diagnoally placed under the micro-sized mirror in order to control the direction which a micro-sized mirror rotates to, via excerting electrostatic forces.

Along a single axis, a single DMD mirror can be modelled as a single slit that introduces an optical path difference to incident light, as a function of the reflected angle. Therefore, a single DMD mirror yields a diffraction pattern that varies with respect to the direction in which the DMD mirror is tilted. The positive direction is defined as the anticlockwise rotation with respect to the DMD surface, and the negative direction is defined as the clockwise rotation with respect to the

DMD surface. The DMD mirrors that are rotated in the positive angle of 12 degrees are regarded as the DMD mirrors that are on their 'On' states. Similarly, the DMD mirrors that are rotated in the negative angle of 12 degrees are regarded as the DMD mirrors that are on their 'Off' state. The optical path differences that are introduced by a DMD mirror in its 'On' state δ_{on} and in its 'Off' state δ_{off} are derived in Appendix A4. The optical path differences δ_{on} and δ_{off} are all functions of the diffraction angle θ_m; and therefore, the intensity of the diffracted light can be also written as a function of the diffraction angle θ_m:

$$I(\theta_m) \sim I_0 \frac{\sin^2\left({}^{\delta(\theta_m)}/_{2}\right)}{\left({}^{\delta(\theta_m)}/_{2}\right)^2} \qquad 2-4$$

Figure 2-6 presents the intensity of the diffracted light for the 'On' DMD mirror as the red line, and the 'Off' DMD mirror as the green line. The equation 2-4 can be derived similarly to the derivation of the 'beating' effect shown in Figure 2-3 (equation 2-2, see Appendix A2), where a series of electric fields all possess a fixed phasor difference ϵ between two adjacent electric fields, and therefore can be written in the form of a geometric sequence (i.e., $E(\cdot)(1 + \exp i\epsilon + \exp i2\epsilon + \cdots + \exp i(N-1)\epsilon)$). In equation 2-2, this fixed phasor difference is $\epsilon = -\Delta\omega t$, whereas in equation 2-4 the fixed phasor difference is $\epsilon = \Delta\delta$. Apart from the optical path difference that is introduced by a single DMD mirror (i.e., δ_{on} and δ_{off}), multiple adjacent DMD mirrors can also introduce an optical path difference $\delta_{multiple}$ to the incident light in a similar manner to a blazed grating (derived in Appendix A4), where at the diffraction angle θ_m that leads to $\delta_{multiple} = m\lambda, m \in \mathbb{Z}^+$, constructive interference is produced. Figure 2-6 presents this diffraction in red solid circles for the 'On' state DMD mirrors and in green solid circles for the 'Off' state DMD mirror. It can be seen from the figure that this diffraction does not have a dependence on the tilt direction of the DMD mirror, and the highest diffraction efficiency (i.e., the 4th diffraction order) is observed only for the 'On' DMD mirror. This enables a DMD to function as a binary intensity mask, which modulates the spatial intensity profile of the incident light in accordance with the geometrical shape of the 'On' DMD mirror array (i.e., all the mirrors on the DMD). Therefore, through controlling the geometrical shape of the 'On' DMD mirror array, the modified light can be imaged onto a workpiece via a microscope objective, allowing the light-matter interaction (e.g., laser ablation) to take place in a specifically chosen region. Notably, the selected values in the simulation of Figure 2-6 are values from the experimental setup in Chapter 5, where the wavelength of the incident pulses is 800 nm, the length of a single DMD mirror is 7.64 μm and the incident angle is approximately 23 degrees. This simulation is based on the *Texas Instruments DLPC3000* micromirror device, where the DMD array is composed of 684 columns by 608 rows micro-sized mirrors that are arranged in a diamond lattice. This simulation is in a relatively good agreement

with the experimental setup, in which pulses from the 4th diffraction order (i.e., the strongest order) are reflected normal to the DMD array surface. A novel application of DMDs for femtosecond laser machining is presented in Chapter 5.

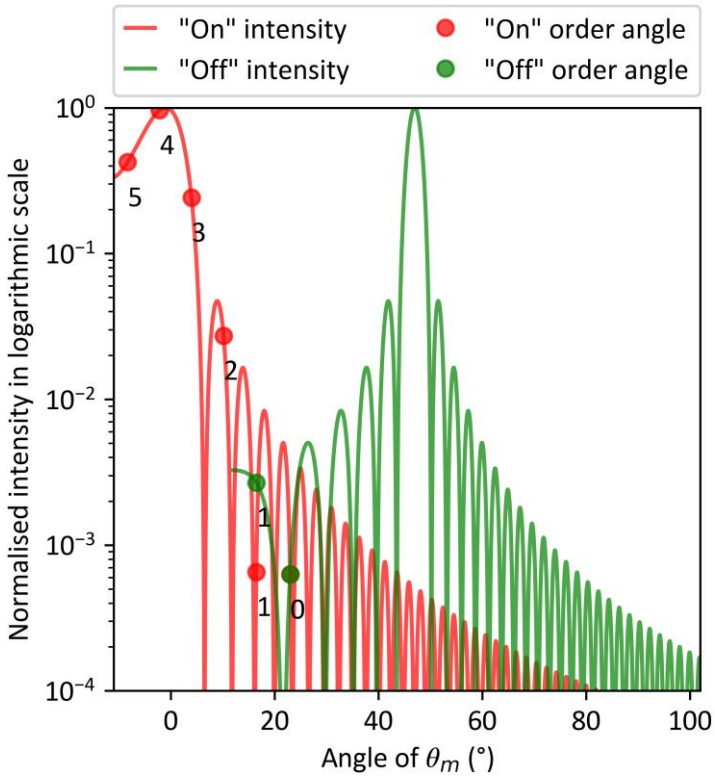

Figure 2-6 Intensity of the diffraction pattern versus the reflected angle; the length of a single micro-sized mirror d is 7.64 µm; the wavelength of the incident light is 800 nm and the incident angle $\theta_i$is 23 degrees. Note that diffractive light that is on the same side of the incident beam, with reference to the normal that is orthogonal to the DMD mirror (i.e., grating surface normal), is physically meaningless, and thus the associated angles are omitted in the figure.

2.3 Mie Scattering

The Mie theory concerns the elastic scattering of light from a perfectly spherical homogeneous object with a smooth surface, whose diameter is comparable to the wavelength of the irradiating light (Bohren and Huffman, 2008). The Mie theory is an exact solution to the Maxwell equations, which in turn allows the electric and the magnetic field inside and outside of a microsphere to be calculated. It usually describes the scattered light from a microsphere in a spherical coordinate system. Specifically, in Mie theory, the transverse components of the electric field that is scattered from a sphere can be written as:

$$E_\theta(r,\theta,\varphi) \sim E(ikr)^{-1} e^{ikr} \cos(\varphi)\, S_2(\cos\theta) \qquad (2-5)$$

$$E_{\varphi}(r,\theta,\varphi) \sim -E(ikr)^{-1}e^{ikr}\sin(\varphi)\,S_1(\cos\theta) \qquad (2-6)$$

where E is the amplitude of the electric field, r, θ and φ are the commonly defined radius, polar angle, and azimuth angle of the spherical coordinate system and k is the wavenumber of the scattered light. The full derivative of Mie theory can be found in (Bohren and Huffman, 2008). S_1 and S_2 are the two complex amplitudes of the scattered electric field along the two orthogonal transverse directions, and they are both functions of the cosine of the polar angle. Figure 2-7 presents a simulation of the intensity of the scattered light from two types of microspheres in vacuum for a wavelength of 650 nm, which corresponds to the laser and microparticle parameters presented in Chapter 4, where a novel application is presented. The code that supports figure 2-7 is a modified version of the *miepython* library, which was developed by Scott Prahl et al, and the original version is available at (Prahl, 2022).

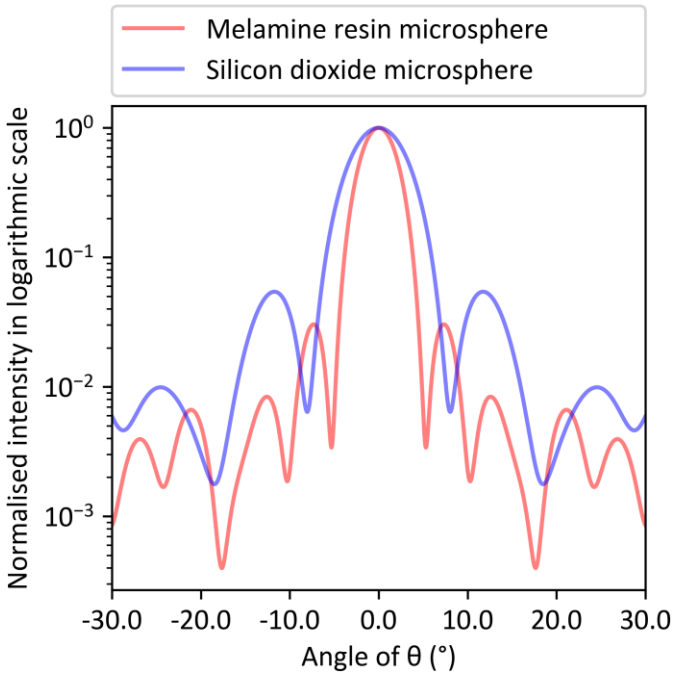

Figure 2-7 Normalised complex amplitude of a melamine resin microsphere (8 µm in diameter, red line) and a silicon dioxide microsphere (5 µm in diameter, blue line) in vacuum along the direction θ. The incident light is a plane wave whose wavelength is 650 nm, and therefore the size parameters (i.e., $^{\pi d}/_{\lambda}$) for these two microspheres are approximately 41.9 and 26.2, which are both smaller than 50, meaning that the Mie theory is applicable to the scattered light of these microspheres.

2.4 Optical Tweezers Effect

Owing to the fact that photons carry momentum (i.e., $p = \hbar k$) and hence the law of conservation of momentum is also applicable to photons, light that is scattered, absorbed, refracted and

reflected by an object also exchanges momentum with the object (momentum is a vector, and scattering, absorption refraction and reflection of an incident light may change its direction). As such, an incident photon is able to exert force to an object if this exchanged momentum varies with time, since force is the time derivative of momentum. In light-matter interactions where the size parameter (i.e., $\pi d/\lambda$) is in excess of 0.1, the reflection and transmission of an incident photon with an object can be approximated via geometrical optics (Gauthier and Wallace, 1995). See for examples (Ashkin, 1992, Hinojosa-Alvarado and Gutiérrez-Vega, 2010), where theoretical calculations of the trapping forces were carried out using geometrical optics for both spherical and non-spherical objects.

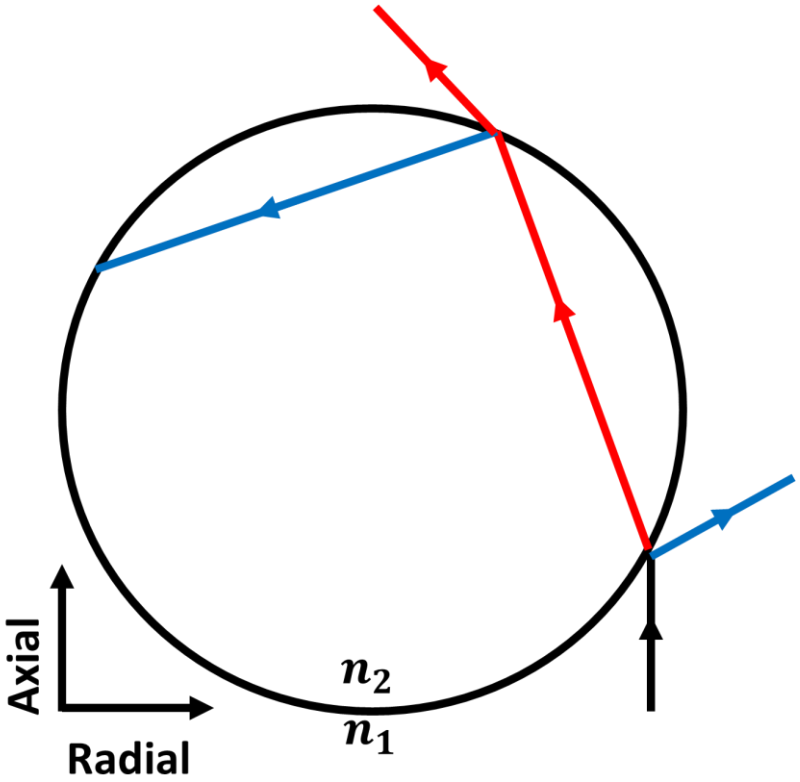

Figure 2-8 Reflected and refracted light rays inside and outside of a prefect sphere, where the reflected light rays are colour-coded in blue, and the refracted light rays are colour-coded in red. Both the reflected and refracted light rays can be decomposed into two component vectors along the axial and the radial directions, which in turn means that it is possible for momentum to be exchanged between the incident photons and the sphere along the radial direction.

In geometrical optics, if the refractive index of an object (whose dimension is larger than the wavelength of the incident light) is different from the refractive index of its surrounding media, multiple reflections and refractions of the incident light inside and at the surface of the object may occur, causing additional resultant light rays. The angle of these rays, with respect to the normal,

will be different from the incident angle. The radial component (i.e., plane orthogonal to the incident light) of the resultant light rays therefore enable optical forces in the radial direction, since momentum is exchanged between the incident light and the object in the radial direction (see Figure 2-8). If the spatial intensity profile of the incident beam is Gaussian distributed in the radial direction, moving the object relative to the incident beam in the radial direction causes the intensities of the resultant light rays to vary, which in turn leads to forces to be exerted in the radial direction since the exchanged momentum in the radial direction starts to vary with time (intensity is proportional to photon density, and therefore higher intensity means higher photon density, which can be translated to more momentum exchanged per unity area). The resultant force could point at or away from the centre of the Gaussian distributed light in the radial direction. Therefore, given the right geometry and refractive index of the object, the right refractive index of the surrounding area, and the right numerical aperture (i.e., $\mathrm{N.A.} = n \sin\theta$., a large N.A. provides a large incident angle), a strong enough attractive force can be exerted to the object in the radial direction when the object is moved away from the incident light in the radial direction, leading to an optical trapping of the object (i.e., the object is immovable with respect to the incident light).

In the axial direction, on the other hand, the scattered (as shown in Figure 2-7) and absorbed light all give rise to an optical force that is exerted to the object along the direction of the incident light (i.e., the axial direction), functioning effectively as a force that sabotages the optical trapping of the object. In certain cases, the reflected and refracted light rays along the axial direction can also participle in this repulsive force. Common solutions that address the repulsive force comprise, for example, a dual-beam system where two incident beams of light are positioned face to face in order to offset the axial forces (or orthogonal to each other so the attractive forces can be applied in two orthogonal directions), and employing gravity or electrostatic force to compensate for the repulsive force along the axial direction (Gauthier and Wallace, 1995). In a less complex setup, a focused Gaussian beam can introduce a Gaussian intensity profile along the axial direction, which enables the attractive force along the axial direction to compensate for the repulsive forces. A novel application of the optical tweezers for the non-contact movement of microparticles is presented in Chapter 6.

2.5 Material Removal with Ultrafast Lasers

Laser material removal processes (or subtractive laser processing) are usually governed by many factors — for a laser source, the contributing factors could be the wavelength, the size of the beam waist, the spatial and temporal profile and the fluence of the incident laser radiations (Kannatey-Asibu Jr, 2009); for a material that is irradiated by laser light, the contributing factors could be the surface roughness (Mustafa et al., 2019), the thickness (Domke et al., 2014) and the band gap

structure of the target workpiece (Kannatey-Asibu Jr, 2009). The laser machining process can also have a significant impact on the material removal. For instance, multiple laser pulse to the same position on a workpiece could introduce an 'incubation' effect (Byskov-Nielsen et al., 2010), which reduces the laser damage threshold of the target material. Or, irradiating the laser to nanostructures (e.g., nano-spheres) could enable their near-field excitations via Mie or Rayleigh scattering (Terakawa and Nedyalkov, 2016), which are able to machine a target workpiece in a resolution that is beyond the diffraction limit of the far-field optics. Specifically, ultrafast lasers are of great importance to the studies of laser materials processing because of their short pulse duration (of the scale of picosecond (10^{-12} s) to femtosecond (10^{-15} s)) and high peak power (of the scale of gigawatt (10^{9} W) to terawatt (10^{12} W)).

The commonly accepted understanding of laser-matter interaction suggests that a pulse with a duration in excess of tens of picoseconds firstly produces a dense plasma layer on the contact area (Rethfeld et al., 2017). The external laser field then transfers its photon energy to the electrons in the plasma layer via the collisional inverse Bremsstrahlung effect or/and via collisionless resonant absorption. The energy from these energised electrons is thereafter transferred to the lattice of the contact area via the Coulomb interaction, and consequently, increases the temperature of the lattice on the contact area. The heat is therefore diffused into the vicinity where the lattice temperature is lower, resulting in a variety of thermal effects on the workpiece. The characteristic relaxation times in the aforementioned processes, namely the time taken for the external laser field to transfer its photon energy to the electrons, the time taken for the electrons to transfer their energy to the lattice and the time taken for the lattice to dissipate heat through thermal diffusion, are usually shorter than the duration of the radiating laser pulse.

On the other hand, the duration of a femtosecond pulse is generally appreciably shorter than these characteristic relaxation times and, as a result, thermal effects are not the dominant mechanism that causes material removal. Instead, the dominant mechanism that is responsible for material removal is the electrostatic ion acceleration that pulls ions away from the contact area (Rethfeld et al., 2017). At the beginning of a femtosecond laser material removal process, impact ionisation and multi-photon ionisation create a dense plasma layer on the contact area. The relaxation times of these ionisation processes and the relaxation time for the photon-electron coupling (i.e., 10^{-17} s) are usually shorter than the duration of a femtosecond pulse This causes the energised electrons and ions to accumulate, owing to the fast photon-electron coupling relaxation time and the slow electron-lattice coupling relaxation time. And therefore, strong electrostatic forces between the electrons and the ions are able to break the ionic bonds, and consequently eject the ions from the workpiece into the air, which causes material removals (Gamaly et al., 2002) (in some areas, this process is known as the 'Coulomb explosion' (Stoian et al., 2000)).

Whilst the duration of a femtosecond pulse is shorter than certain characteristic relaxation times, it does not, however, imply that the associated thermal effects (e.g., thermal shock and thermal diffusion) do not happen in a femtosecond laser-matter interaction in practice (Gattass and Mazur, 2008). However, it is evident that the thermal effects and the consequential collateral damages are considerably less significant in the case of femtosecond laser-matter interactions than that experienced with nanosecond and picosecond laser-matter interactions. In addition, the damage threshold of a bulk, when it is irradiated by a femtosecond laser, is less material-dependent (i.e., metallic or dielectric) and more intensity-dependent. Therefore, the femtosecond laser pulses can enable the machining of finer structures, and with less collateral damage, than lasers with longer pulse durations (Gamaly et al., 2002). In this thesis, novel applications for femtosecond laser ablation are demonstrated in Chapter 5 and Chapter 7.

2.6 Summary

As discussed in this chapter, the interaction between light and matter can be non-destructive or destructive, generally based on the light intensities used. In the case of low intensity, incident light can be scattered by the medium, resulting in a specific angular distribution of scattered light (which can be described using Mie scattering theory in specific cases), and a change in momentum can result in the change of position of the medium (i.e., the optical tweezers effect). In the case of high intensity, the incident light can result in removal of the surface of the medium, where the nature of material removal depends on a wide range of parameters, such as the laser wavelength and pulse length, and the material refractive index and damage threshold. The interaction of light and matter is regarded as extremely complex, particularly for very high intensities. This leads to the one of the primary objectives of this thesis, namely to demonstrate the application of deep learning for assisting with the modelling, understanding, and indeed control, of the interaction of light and matter. Deep learning is introduced in Chapter 3.

Chapter 3 Deep Learning

The application of algorithms for assisting in scientific research has a long history. In recent years, this application has become considerably more noticeable, as there has been a rapid move to apply a novel range of powerful 'artificial intelligence' algorithms, also known as deep learning, to scientific research. These deep learning algorithms offer the potential to develop an understanding and optimisation of complex scientific research challenges, which otherwise in many cases may be too complex to solve. Deep learning is a 'data driven' scientific technique, where an understanding of a complex system can be achieved through the algorithmic processing (i.e., neural network training) of experimental data, and as such, the challenge is often the question of which sets of data to collect (and how to collect this data) for training of the deep learning approaches. In this chapter, an overview of deep learning is provided, starting from machine learning, artificial neural networks, and convolutional neural networks (relevant for Chapter 4 and Chapter 5), followed by a discussion of virtual training environments and reinforcement learning (relevant for Chapter 6 and Chapter 7).

It is important to realise that historically the fields of photonics and computer science generally had minimal need for a rigorous overlap, and therefore simultaneous expertise across both research domains is rare. However, given the increasing prevalence of deep learning across all research disciplines, awareness of the theoretical foundations of deep learning is becoming increasingly important for non-computer scientists. Understandably, the contents of this chapter may be challenging for some researchers with sole expertise in photonics. However, whilst this introductory chapter has a high level of computer science technicality, the experimental research chapters are generally accessible to researchers with minimal deep learning experience. As such, comprehension of the contents of this chapter is not a requirement for understanding the four experimental chapters (chapter 4, 5, 6 and 7). The reader may therefore wish to firstly glance through this introductory chapter, and then return to learn more about specific topics as and when needed.

3.1 Machine Learning and Supervised Learning

Machine learning is a collection of computer algorithms that rely on data-driven processes to perform certain tasks. Compared to conventional algorithms, which are generally reliant on human expertise and programmatical descriptions to construct solutions, a machine learning algorithm learns directly from data from the system to be solved. In other words, rather than manually constructing the internal processes that provide the solution to a designated task, the solution is determined via 'a training process', where the machine learning parameters are automatically

tuned through processing the data. It is first assumed that an internal function $y = f(x)$ is parameterised by θ (i.e., $y = f(x;\theta)$) and a desired target function T is described through mapping inputs X to outputs Y (i.e., $T: X \to Y$). A conventional algorithm generally relies on a programmatical description to find the optimal parameter θ to realise $Y = f(X;\theta)$, whilst a machine learning algorithm utilises a set of data $E = \{(x_i, y_i)|x_i \sim X, y_i \sim Y, i = 0,1,2,3,\cdots,N\}$ to infer a parameter θ that realises $y_i = f(x_i;\theta)$. Therefore, if it is not possible to determine the parameter θ directly, an approximate solution can instead be obtained via a machine learning algorithm. Note here that a parameter θ is referring to a collection of all parameters inside a neural network, not a single parameter inside the neural network. The most accepted definition of machine learning comes from Mitchell(Mitchell, 1997):

> *'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E'*

Amongst the wide spectrum of machining learning studies, the neural network is arguable one of the most successful and popular architectures. An artificial neural network can be regarded as a parameterised function approximator that is possible to realise $Y = f(X;\theta)$) through training with data E. Depending on whether a set of ground truth is included in the data E, there exist two types of learning, namely supervised learning, and unsupervised learning. The supervised learning of an artificial neural network can be decomposed into two fundamental processes, namely the feedforward and the backpropagation (Goodfellow et al., 2016), which is discussed further in Section 3.2. Of particular interest in the study of the artificial neural network is a variant commonly known as the Convolutional Neural Network (CNN). The convolutional neural network is a specific type of artificial neural network that is optimised for processing visual signals (e.g., camera observations). That is, the convolutional neural network is efficient and effective in processing two-dimensional input $Y \in \mathbf{O}^{m \times n}$, where m and n are the dimensions of the input Y and $\mathbf{O}$ is the set of permitted values (e.g., commonly, images are stored in 8-bits integers, which means that $\mathbf{O} = \{0,1,2,\cdots,255\}$). Note that one-dimensional and three-dimensional signals are also applicable to the convolutional neural network, provided that dimension of the filters in the convolutional neural network are changed accordingly. However, these cases are less common and are not within the scope of the studies in this thesis, and hence will not be further discussed. The convolutional neural network is discussed in Section 3.3.

It is worth noting that machine learning algorithms are not necessarily superior to conventional algorithms. In some research areas, machine learning algorithms can perform less effectively than that of conventional algorithms (Tatarchenko et al., 2019). In addition, owing to the

data-driven nature, in some cases, it is sometimes impracticable or even unviable to employ a machine learning algorithm.

3.2 Artificial Neural Network

An artificial neural network can be understood via the two intertwined processes involved in the training of an artificial neural network (abbreviate here to neural network), namely the feedforward process and the backpropagation process. Again, let $y = f(x)$ be a neural network parameterised by θ, whose training is aimed to approximate a function T that can be described through mapping input X to output Y (i.e., $T: X \to Y$) via the training material $E = \{(x_i, y_i)|x_i \sim X, y_i \sim Y, i = 0,1,2,3, \cdots, N\}$. The feedforward process takes the inputs $\mathcal{X} = \{x_i|x_i \sim E, i = 0,1,2,3, \cdots, N\}$ from the training material E and maps them to a set of 'predictions' through inputting $\mathcal{X}$ to the neural network $\mathcal{Y}_{\text{predict}} = \{\hat{y}_i|\hat{y}_i = f(x_i; \theta)\ , i = 0,1,2,3, \cdots, N\}$. If the parameter θ in the neural network is not carefully tuned towards mimicking the target function $T: X \to Y$, it is likely that the set of predicted outputs $\mathcal{Y}_{\text{predict}}$ will have large discrepancies compared to the ground truth with respect to the input $\mathcal{Y}_{\text{label}} = \{y_i|y_i \sim E, i = 0,1,2,3, \cdots, N\}$ from the training material E. Therefore, an optimisation process is needed to fine-tune the parameter θ in order for the neural network to approximate the target function T. The backpropagation process is usually adopted in conjunction with the feedforward process to find a parameter θ for the neural network to approximate the target function T. A performance measurement function $P(\mathcal{Y}_{\text{predict}}, \mathcal{Y}_{\text{label}})$ (more commonly known as an 'objective function') is generally used in order for the backpropagation process to determine these discrepancies in the form of numerical values, and the parameter θ in the neural network $y = f(x; \theta)$ can consequently be inferred towards the direction of minimising the determined values from the performance measurement function P, via for instance, the gradient descent method or the Newton's method. If, through repeating this training step (i.e., an instance of the feedforward followed by an instance of the backpropagation), a good parameter θ_{good} is found, which enables the neural network $y = f(x; \theta_{\text{good}})$ to function similarly to the target function T, at least on the training material E, this neural network $y = f(x; \theta_{\text{good}})$ can be then used for its intended purposes.

Theoretical studies demonstrate that a neural network possesses an excellent ability to approximate other functions. Specifically, the universal approximation theorem suggests that (Goodfellow et al., 2016):

> *'A feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function (such as the logistic sigmoid activation function)*

can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units.'

However, this promising theoretical upper bound is hard to achieve in practice. Theoretical studies generally indicate that the successful training of a neural network requires a large number of training material E. The primary concern is that since E is a subset of X and Y, the neural network that performs well on E (i.e., $y_i = f(x_i; \theta_{\text{good}})$) cannot be guaranteed to also perform well on X and Y. The discrepancy is usually described as the 'generalisation error', where this error can be proven to be upper bounded by a function that is inversely proportional to the size of E (Sontag, 1998). That is, a larger dataset leads to a better performance of the neural network on unseen data. In Chapter 5, a data augmentation method is proposed in order to expand the size of a laser-machining dataset. Specifically shown in Figure 5-6 is the impact of the size and the richness of the dataset on the effectiveness of the neural network. In Chapter 6 and Chapter 7, it is shown that the neural network can be trained in a virtual environment that approximates the corresponding physical environment, in order to minimising the reliance on real-world data collection. The relationships between these concepts are highlighted in Figure 3-1.

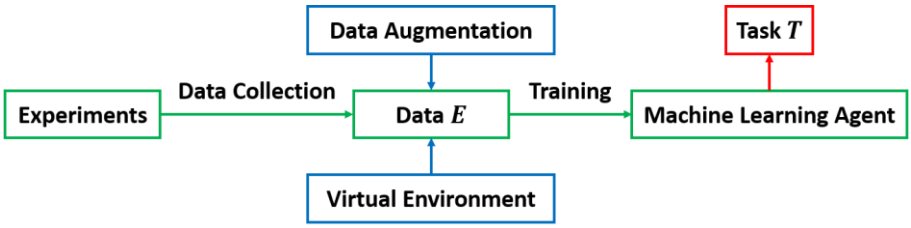

Figure 3-1 Data augmentation and virtual environment reduce the reliance on the data that are collected from real-world experiments.

3.2.1 Feedforward Process

A neural network is commonly described as a hierarchy of interconnecting differentiable functions. Owing to the hierarchical system of connectivity, a neural network is often interpreted as a function that possesses a number of layers. Conveniently, these layers can be categorised into three types, namely the input layer (in some literature it is known as the 'visible layer'), the hidden layer, and the output layer. Generally speaking, the input to the input layer and the output of the output layer are interpretable for humans (Goodfellow et al., 2016). For instance, the input to an input layer could be a sensory signal that takes the form of a photograph of a cat, and the output of the corresponding output layer could be the classification of the cat breeds (e.g., 90 % confidence as British shorthair and 10 % confidence as American shorthair). The outputs of hidden layers, on the other hand, are generally not interpretable by humans, and is commonly described as the abstract

representations of the inputs. Figure 3-2 depicts the connectivity between the input layer, the hidden layer, and the output layer. Whilst this discussion presents a useful concept for understanding neural networks, it should be clarified that the inputs and outputs of some more-complex neural network systems are not necessarily interpretable for human (e.g., outputs of an auto-encoder (Kramer, 1991)).

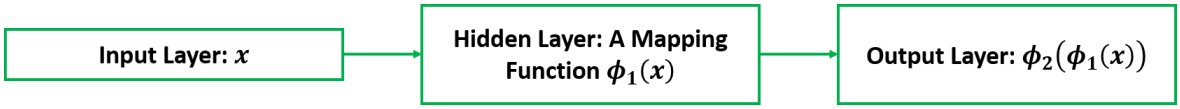

Figure 3-2 Schematic of the feedforward process: an input x is mapped to an output $\phi_2(\phi_1(x))$ through two mapping functions ϕ_1 and ϕ_2.

In a simple neural network, where the input and the output of the neural network are all of the form of a one-dimensional matrix, the connectivity of a feedforward process can also be interpreted via matrix multiplication, as shown in Figure 3-3. In this figure, an input matrix x of shape n can be understand as, for example, a series of numerical measurements of a physical process (which are abstract but are nevertheless interpretable for humans). This input is multiplied by a matrix of shape m by n, and is consequently transferred to a one-dimensional matrix $\phi_1(x)$ of shape m. Note here that each element $x_{2,h}$ in the resultant matrix $\phi_1(x)$ (i.e., the output of the hidden layer) is the dot product between all elements in the input matrix $x = [x_{1,1}, x_{1,2}, \cdots, x_{1,n}]$ and the corresponding parameter vector ${k_h}^T = [k_{h,1}, k_{h,2}, \cdots, k_{h,n}]^T, 1 \leq h \leq m, h \in \mathbb{Z}^+$, and this dot product can be written as $x_{2,h} = x{k_h}^T = x_{1,1} \cdot k_{h,1} + x_{1,2} \cdot k_{h,2} + \cdots + x_{1,n} \cdot k_{h,n}$. After this multiplication, the output $\phi_1(x)$ is presumably not interpretable to humans. In order to transfer this abstract representation of the input back to a one-dimensional, interpretable matrix (which corresponds to, for example, certain attributes of the aforementioned physical process), an output layer with a parameter matrix of the shape t by m is multiplied to the abstract representation of the input (i.e., $\phi_1(x)$), yielding an interpretable matrix of shape t ($\phi_2(\phi_1(x))$).

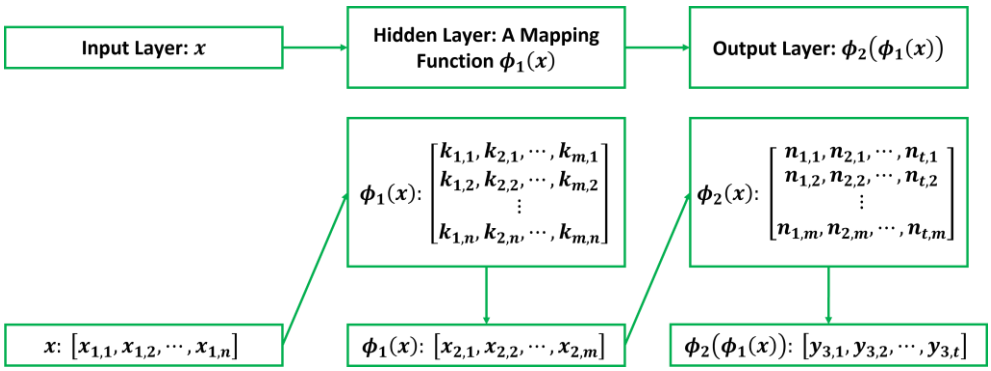

Figure 3-3 Feedforward process in the form of matrix multiplications.

The computation inside a layer of a neural network is commonly described as a linear function followed by an applied non-linear activation function as shown in equation 3-1, where x is the input matrix, k and b perform an affine transformation on the input $k \times x + b$ and a non-linear function $g(x)$ is applied to the transformed matrix. This computing process is differentiable, and contains learnable hyperparameters (i.e., k and b). This activation function $g(x) = \mathrm{Max}(0, x)$ is called Rectified Linear Unit (ReLU). Other common activation functions include, for example, logistic sigmoid $g(x) = (1 + e^{x})^{-1}$ and hyperbolic function $g(x) = \tanh(x)$.

$$\begin{aligned} \phi(x) &= g(k \times x + b) \\ x &= [x_1, x_2, \cdots, x_n] \\ k &= \begin{bmatrix} k_{1,1}, k_{2,1}, \cdots, k_{m,1} \\ k_{1,2}, k_{2,2}, \cdots, k_{m,2} \\ \vdots \\ k_{1,n}, k_{2,n}, \cdots, k_{m,n} \end{bmatrix} \\ b &= [b_1, b_2, \cdots, b_m] \\ g(x) &= \mathrm{Max}(0, x) \end{aligned} \qquad (3-1)$$

This connectivity of the feedforward process is loosely inspired by the biological structure of a neuron (Goodfellow et al., 2016). It is believed that this 'connectionism' of the feedforward enables learning and memory abilities of a biological neuron. Similar to how a stimulus propagates through neurons, in a feedforward process an input vector is able to propagate through a number of the non-linearised linear functions, as shown in Figure 3-4. Of particular interest is the aforementioned activation function ReLU — in the cortical system of an animal, the responses of a single neural to stimuli can be both digital and analogue (Dahl et al., 2013). Whilst a neuron can focus on a single stimulus and ignore other stimuli (i.e., digital), the response of the neuron to the focused stimulus can be linear (i.e., analogue). This idea of a non-linear function that possesses both the analogue and digital properties was first proposed in silicon circuit design that imitates cortical circuits (Hahnloser et al., 2000), but was later introduced to the studies of artificial neural network. Technically speaking, the ReLU function $g(x) = \mathrm{Max}(0, x)$ is not differentiable at $x = 0$, since on the left side of $x = 0$ the derivative is 0, whilst on the right side of $x = 0$ the derivative is 1. This discontinuity in derivative is usually omitted because it occurs only at $x = 0$. Due to the connectivity, the layers shown in Figure 3-4 are commonly named 'Fully Connected Layer' (FCL), which are usually characterised by the size of its output. For instance, the hidden layer in this figure is of the size m.

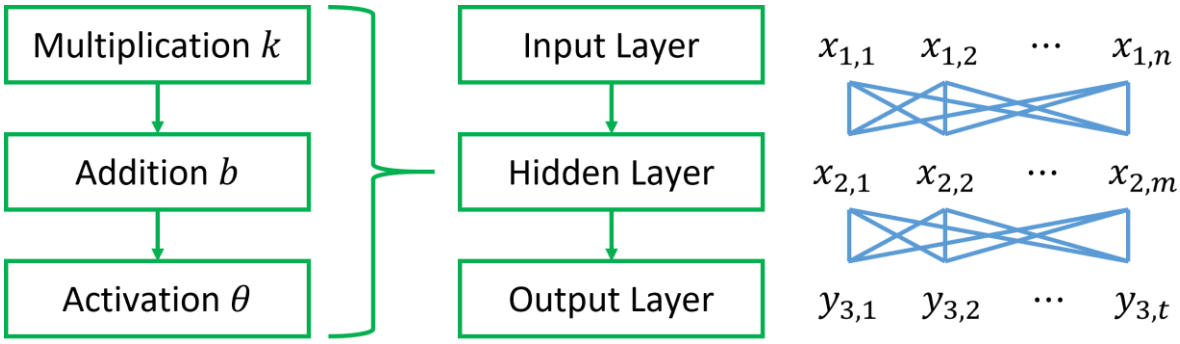

Figure 3-4 Connectionism of the feedforward process. This type of layer is more often referred to as the fully connected layer.

3.2.2 Backpropagation Process

In order to find a suitable parameter θ that can enable a neural network to approximate a target function T, the backpropagation process (Rumelhart et al., 1986) usually computes the gradients of all learnable parameters with respect to a performance measure function P, and through employing, for example, the gradient descent method, the parameter θ can be moved towards the direction of minimising the discrepancy that is determined by the performance measure function P, as shown in Figure 3-5.

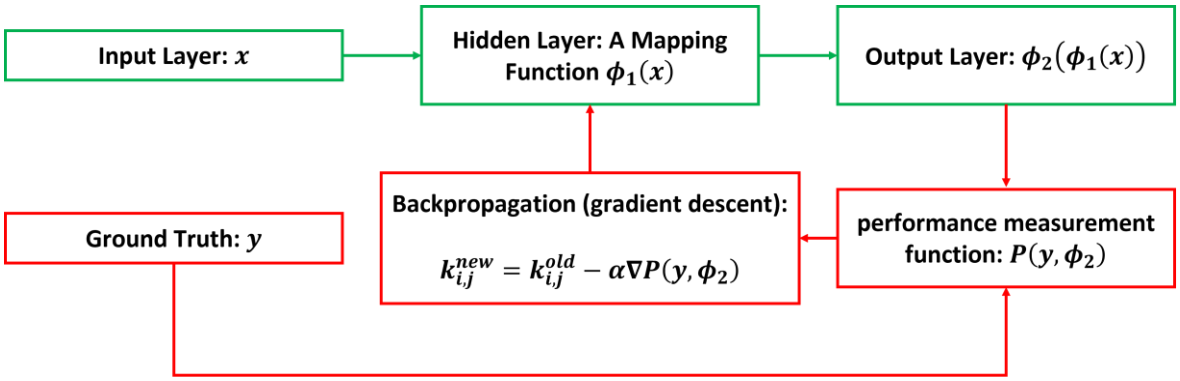

Figure 3-5 Schematic of the backpropagation process: the hyperparameters inside the hidden layer are tuned towards minimising the discrepancy between its output and the ground truth, which is measured by the performance measurement function. The α in the gradient descent process is a hyperparameter commonly known as the learning rate, which controls the step size of each optimisation iteration.

The choice of the performance measurement function (referred to as the objective function) depends on the functionality of the neural network. For instance, in a neural network that functions as a classifier (i.e., categorical identification of the associated inputs), the adopted objective function usually measures the relative entropy of two probability distributions (i.e., the Kullback-Leibler divergence, which measures the similarity between two probability distributions). For the research presented in this thesis, the neural networks are primarily used to map inputs to outputs

in a continuous range (i.e., continuous real number). To some extent, it can be said that the neural networks presented in this thesis all concern the regression problem, whose objective function can be given by in the form of the Mean Square Error (MSE):

$$MSE = \frac{1}{2} \cdot \frac{\sum_{t=1}^{N} (Prediction_t - GroundTruth_t)^2}{N} \qquad (3-2)$$

This objective function is different than the commonly defined mean square error, in the sense that a constant $1/2$ weights the mean square error. This constant is purposely added to the equation in order to ensure that the derivative of this objective function is not weighted by the power of 2. The partial derivative of the equation 3-2 with respect to a parameter k of the hidden layer can be given by:

$$\frac{\partial MSE}{\partial k} = \frac{1}{N} \sum_{t=1}^{N} (Prediction_t - GroundTruth_t) \cdot \left(\frac{\partial Prediction_t}{\partial k} \right) \qquad (3-3)$$

This objective function is sometimes referred to as the Half Mean Square Error (HMSE).

The partial derivative of an objective function with respect to a parameter k in the hidden layer follows the chain rule in partial differentiation. In Figure 3-3, an input vector $x = [x_{1,1}, x_{1,2}, \cdots, x_{1,n}]$ is sequentially multiplied by ϕ_1 and ϕ_2, which eventually yields an output vector $y = [y_{3,1}, y_{3,2}, \cdots, y_{3,t}]$. For the following, it is assumed that the HMSE objective function is adopted, which measures the discrepancy between the output vector and a vector of ground truth $y_{\mathrm{GroundTruth}} = [y_{1,1}, y_{1,2}, \cdots, y_{1,t}]$, and the partial derivative of this HMSE objective function with respect to the parameter $k_{1,1}$ is of interest. According to the equation 3-3, the partial derivative can be written as:

$$\frac{\partial HMSE}{\partial k_{1,1}} = \frac{1}{t} \sum_{j=1}^{N=t} \left(y_{1,j} - y_{3,j} \right) \cdot \left(\frac{\partial y_{3,j}}{\partial k_{1,1}} \right)$$

Since each element in the output y (i.e., $y_{3,j}$) is connected to all the elements in the output of the previous layer $\phi_1(x) = [x_{2,1}, x_{2,2}, \cdots, x_{2,m}]$, i.e., $y_{3,j} = n_{j,1} \cdot x_{2,1} + n_{j,2} \cdot x_{2,2} + \cdots + n_{j,m} \cdot x_{2,m}$, and only $x_{2,1}$ has dependence on the $k_{1,1}$, the partial derivate of $\partial HMSE / \partial k_{1,1}$ can therefore be written as:

$$\frac{\partial HMSE}{\partial k_{1,1}} = \frac{1}{t} \sum_{j=1}^{N=t} \left(y_{1,j} - y_{3,j} \right) \cdot \frac{\partial y_{3,j}}{\partial x_{2,1}} \cdot \frac{\partial x_{2,1}}{\partial k_{1,1}}$$

It is known from the figure that $\partial y_{3,j} / \partial x_{2,1} = n_{j,1}$ and $\partial x_{2,1} / \partial k_{1,1} = x_{1,1}$, and therefore the simplest expression of this derivative can be given by:

$$\frac{\partial HMSE}{\partial k_{1,1}} = \frac{1}{t} \sum_{j=1}^{N=t} (y_{1,j} - y_{3,j}) \cdot n_{j,1} \cdot x_{1,1}$$

Note that both the bias b and the activation function $g(x)$ are not considered in this simplified evaluation of the partial derivative of the objection function with respect to the parameter, and instead they are purposely omitted for the clarity of the equations. In practice, both the bias b and the activation function $g(x)$ are critical to the function approximating ability of the neural network, and both the bias b and the activation function $g(x)$ are frequently presented in the mathematical expressions of the layers in the neural network, and therefore both the bias b and the activation function $g(x)$ must be included in the partial derivatives of the objective function with respect to the parameter if they are part of the neural network.

Accessing the derivative information allows the backpropagation process to change the numerical values of the parameter θ in the neural network in the direction of minimising the determined discrepancies by the objective function, via the gradient descent method or the Newton's method. These optimisation methods belong to an on-going area of research named 'convex optimisation', where algorithms are proposed to find the global minimum effectively and efficiently in the parameter space of an objective function. All the neural network presented in this thesis were trained with the ADAptive Moment estimation (ADAM) optimiser, and the details of the ADAM optimiser can be found in (Kingma and Ba, 2014).

3.3 Convolutional Neural Network

In real world applications, the collected signals that describe an object or a process are often multi-dimensional. For instance, the three-colour-channels camera observation of a laser-machined structure (e.g., Figure 5-5) is a three-dimensional signal, whereas the surface profile of a laser-machined structure (e.g., Figure 5-4) can be regarded as a two-dimensional signal. In addition, the information that these multi-dimensional signals possess is often spatially or temporally correlated.

This can be better illustrated via the two-dimensional Fourier transform of an image, as shown in Figure 3-6; the high-frequency (thus short-wavelength) components of an image usually contains more detailed information, as compared with that of the low-frequency components. These multi-dimensional signals can nevertheless be mapped via neural networks with a 'fully connected' style connectivity, which is previously discussed in Section 3.2.1 and shown in Figure

3-4, provided that any multi-dimensional signal can be reshaped into a single-dimensional signal. However, this style of connectivity fundamentally treats the information that is stored in the spatial or temporal distribution of the signal as a learnable feature. In other words, the 'fully connected' connectivity may be highly inefficient and ineffective in processing a multi-dimensional signal with localised features, since information stored in the spatial or temporal distribution of the signal needs to be learnt separately.

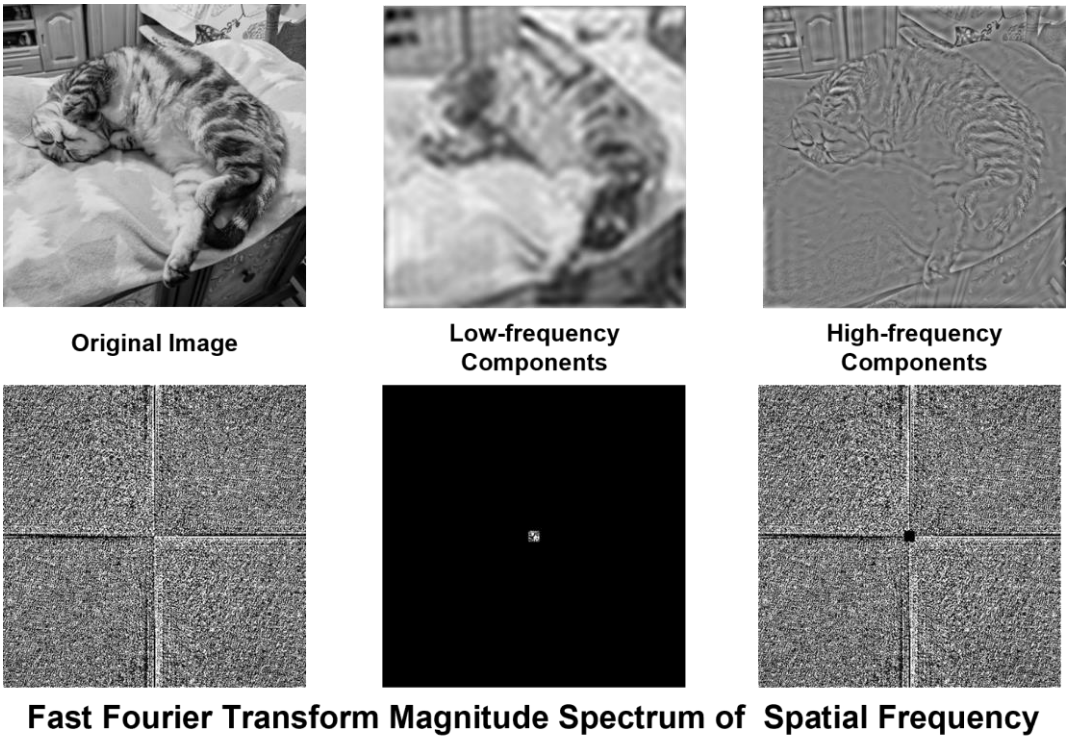

Figure 3-6 Original, low-frequency components and high-frequency components of the test image, are shown in the first row, and their correspinding magnitude spectrums in the spatial frequency domain are shown in the second row. Photo by Yunhui Xie.

Loosely inspired by the mammalian vision, the Convolutional Neural Network (CNN) is a variant of the artificial neural network that is capable of effectively and efficiently processing two-dimensional visual signals (LeCun et al., 1998). In a convolutional neural network, a number of 'filters' scan through a two-dimensional image row-by-row and column-by-column, similarly to the convolution process in signal processing. Each filter produces a resultant 'feature map' of the two-dimensional image. Figure 3-7 presents the depiction of the convolutional layer in a convolutional neural network, where the input array is of the shape 3 × 3 and the filter is of the shape 2 × 2. Similar to the mathematical expression for a fully connected neural network in equation 3-1, the convolutional layers compute the dot product between the learnable filter $k^{m\times n}$ and the input image $x^{q\times p}$, followed by adding a bias b and applying an activation function $g(x)$ to the dot product:

$$y_{a,b} = \sum_{a=1}^{N=q-m+1} \sum_{b=1}^{N=p-n+1} \sum_{i=1}^{N=m} \sum_{j=1}^{N=n} \left(g\left(k_{i,j} \cdot x_{a+i,b+j} + b \right) \right) \qquad (3-4)$$

Note that in signal processing, equation 3-4 is often referred to as the 'cross-correlation', and the convolution process in a two-dimension space is more commonly defined in the form of $\sum_{a=m+1}^{N=q} \sum_{b=n+1}^{N=p} \sum_{i=1}^{N=m} \sum_{j=1}^{N=n} k_{i,j} \cdot x_{a-i,b-j}$, which in the context of Figure 3-7 means the dot product should be given by $k_{1,1} \cdot x_{a+1,b+1} + k_{1,2} \cdot x_{a+1,b} + k_{2,1} \cdot x_{a,b+1} + k_{2,2} \cdot x_{a,b}$. This minor difference is usually omitted since the parameters inside the filter $k_{i,j}$ are learnable. Therefore, it can be seen as a difference in the naming convention.

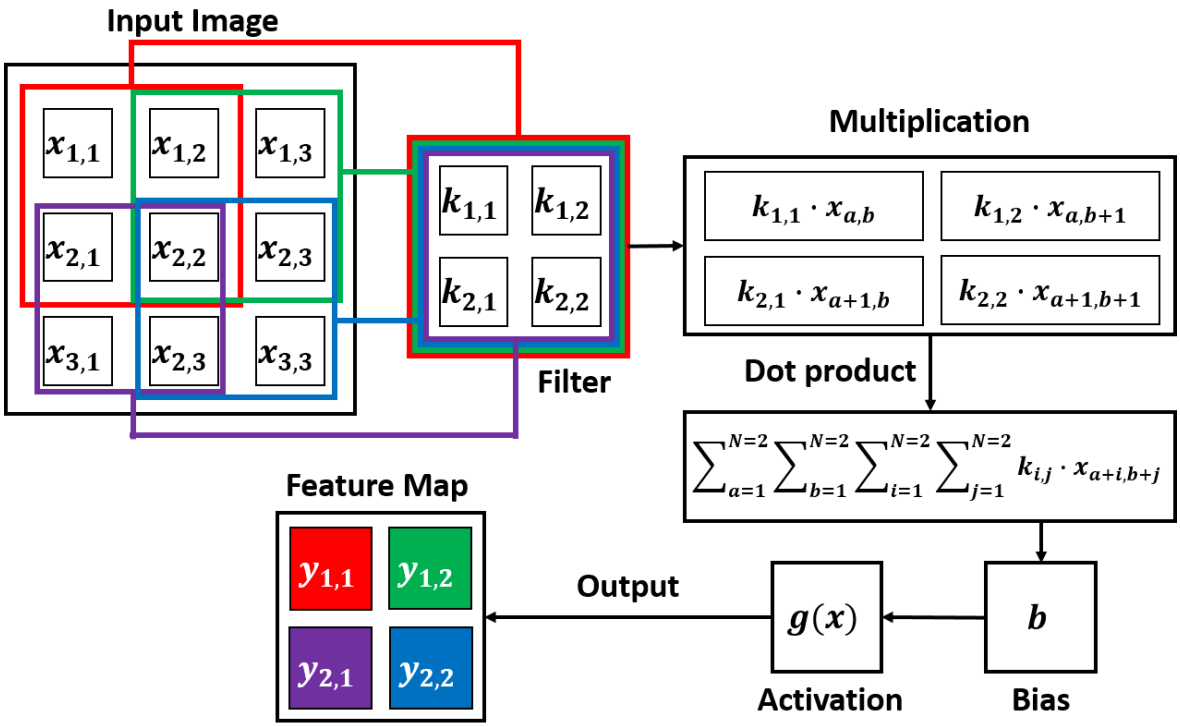

Figure 3-7 Feedforward process of the convolutional layer in the form of matrix multiplications.

The main differences between the convolutional neural network (i.e., Figure 3-7) and the fully connected neural network that (i.e., Figure 3-3) are as follows (LeCun et al., 1998):

1) Local receptive field (sometimes referred to as the 'sparse connectivity').
2) Parameter sharing (sometimes referred to as the 'weight replication' or 'shared weight').
3) Subsampling (sometimes referred to as the 'pooling').

The local receptive field is a style of sparse connectivity that restricts the connections between two neighbouring layers based on the spatial distances of the individual elements between the upper and the lower layer. Comparatively, in the fully connected network presented in Figure 3-3, each element $x_{2,h}$ in the output of the hidden $\phi_1(x)$ is the dot product of all the elements from

the input layer $x = [x_{1,1}, x_{1,2}, \cdots, x_{1,n}]$ and its associated parameter vector ${k_h}^T = [k_{h,1}, k_{h,2}, \cdots, k_{h,n}]^T$, in the sense that $x_{2,h} = x{k_h}^T = x_{1,1} \cdot k_{h,1} + x_{1,2} \cdot k_{h,2} + \cdots + x_{1,n} \cdot k_{h,n}$ (hence the name 'fully connected layer'). However, in the convolutional layer presented in Figure 3-7 and shown in equation 3-4, each element in the output of the convolutional layer is only the dot product of selective elements from the input image. For instance, in the context of Figure 3-7, the element $y_{1,1}$ in the output of the convolutional layer is only connected to four elements that are spatially adjacent to $y_{1,1}$ from the previously layer, namely $x_{1,1}$, $x_{1,2}$, $x_{2,1}$ and $x_{2,2}$ (i.e., elements within the red rectangle). From a different perspective, the information stored in the element $x_{1,2}$ in the input image is only accessible for the elements $y_{1,1}$ and $y_{1,2}$ in the output of the convolutional layer (i.e., red and green rectangles). This sparse connectivity between layers forces the convolutional neural network to learn from localised features from the input signals, and therefore the convolutional neural network is usually considered more effective and efficient in processing two-dimensional real-world inputs.

The parameter sharing of the convolutional neural network is the deliberate re-use of the same set of parameters. In the context of Figure 3-7, this can be understood as the same filter scanning through the input image row-by-row and column-by-column; in the context of the equation 3-4, this can be understood as q>>m and p>>n (i.e., the size of the input image q by p is much greater the size of the filter m by n). The local receptive field and the parameter sharing of the convolutional neural network ensures that the convolutional neural network can utilise much fewer parameters to process a two-dimensional input where features of interest are localised more effectively and efficiently, compared to that of the fully connected layer. For instance, in a fully connected layer, in order to map an input of shape q by p to an array of shape k, a total number of $q \times p \times k$ parameters are required, whereas in a convolutional neural network, in order to map the same input to t feature maps with t filters of the shape m by n, a total number of $m \times n \times t$ is needed. Provided that q>>m and p>>n, if the total number of the resultant feature maps (i.e., the total number of the filters) is not excessively larger than the total number of the outputs in the fully connected layer (i.e., t>>k, which is usually not true in modern neural network designs), the total number of parameters in a convolutional neural network is usually much smaller than the total number of parameters in a fully connected layer. This can be more visually observed in the designs of the neural networks shown in Table 4-1, Table 4-2, Table 5-1, Table 5-2, Table 6-1, Table 6-2 and Table 7-1 in the experimental chapters in this thesis, where the total numbers of parameters for the convolutional neural networks hardly exceed tens of thousands, whilst the total numbers of parameters for other fully connected layers are generally greater than several millions.

The subsampling, which is more commonly known as the pooling layer, typically reduces the size of a feature map via a process that is similar to the convolutional layer (i.e., Figure 3-7), where

instead of performing a dot product between the local receptive field of the input image and the learnable parameters, the local receptive field of the input image is instead applied with a maximum operator (i.e., $\mathrm{Max}(x_{a,b}, a = 1,2,\cdots,m\ b = 1,2,\cdots,n)$) or an averaging operator (i.e,. $\frac{1}{m \cdot n}\sum_{a=1}^{N=m}\sum_{b=1}^{N=n} x_{a,b}$). Additionally, both the bias and the activation are discarded in the pooling process. This pooling layer is usually described as a feature extraction process that summarises the key information from the previous layer into feature maps with a much smaller size. Examples of both the max pooling and the average pooling are shown in Figure 3-8.

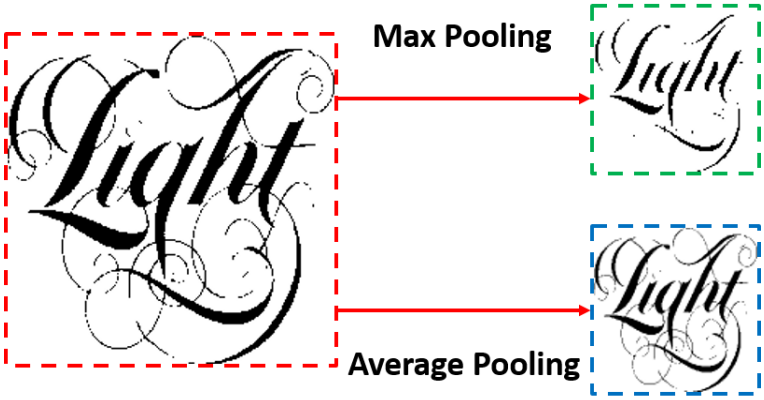

Figure 3-8 Max pooling and average pooling of the same image with a filter of shape 2 by 2.

Typically, both the convolutional layer and the pooling layer are characterised by the attributes of their filters, namely the size of the filter, the number of filters, and the strides that the filter moves per sampling. For instance, the convolutional layer depicted in Figure 3-7 possesses 1 filter of the size 2×2 that moves at a stride of 1. A typical connectivity of convolutional neural network is presented in Figure 3-9, where the features on the raw input are extracted via two convolutional neural networks, each followed by a pooling layer. The extracted features are then mapped to a fully connected layer, which is connected to an output layer. The backpropagation process of a convolutional neural network works similar to that of the fully connected network, where the partial derivative of the objective function with respect to each learnable parameter can be obtained using the equation 3-4 via the chain rule in partial differentiation.

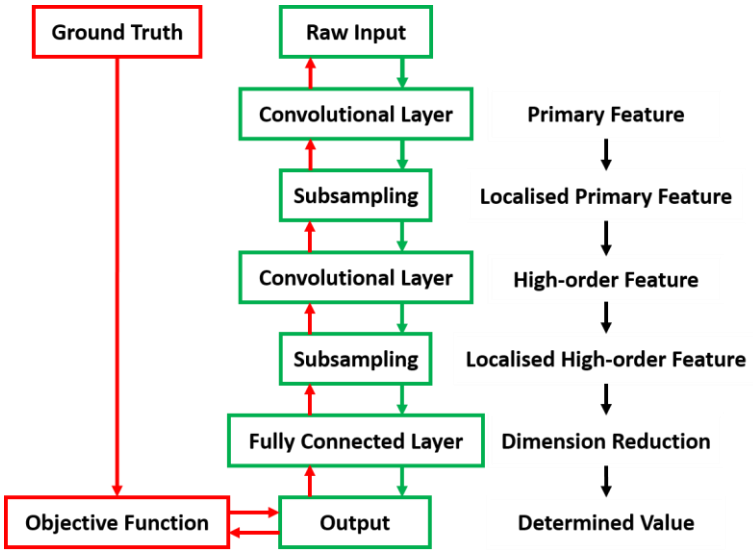

Figure 3-9 Schematic of a convolutional neural network, where the green arrows are the feedforward process, and the red arrows are the backpropagation process.

3.4 Markov Decision Process and the Virtual Training Environment

Supervised learning generally requires labelled data to train a neural network. This means that in order to train a neural network to solve an experimental control problem as a controller, there is a requirement that the ideal course of action to this control problem must be known prior to the training, since the correct labelled data will be needed for the training. A workaround to this limitation is that the neural network can be instead used to determine the critical attributes for this control process, and another decision-making agent (e.g., Model predictive control (Bollig et al., 2003) or a PID controller) can determine the best course of action by leveraging these critical attributes. An example of this control scheme is discussed in detail in Section 5.7, where the convolutional neural network is used to control the laser-machining depth of a structure, through automatically ceasing laser-machining at a specific point in time (with the decision made in real time during laser-machining). In this experiment, instead of directly deducing the action, the convolutional neural network only determines an attribute that indicates the machining depth, and an exterior if-else statement deduces the actual action (i.e., apply additional laser pulses, or apply no more laser pulses) through assessing if this determined attribute is in excess of certain predefined values.

It is important to realise that, in order to ensure that a control scheme can be applicable to a control problem, a generalised description concerning this control problems must be determined first. In the context of this thesis, two control problems are discussed in Chapter 6 and Chapter 7, where, in both cases, the interactions between the controllers and the experimental apparatuses are advanced in discrete time steps. These experiments are therefore framed as the Markov

Decision Processes (MDPs). A Markov decision process can usually be divided into two components, namely a decision-making agent, and an environment with which the decision-making agent interacts. The environment can be usually characterised by four attributes, namely state set S, action set A, state-transition probability P and reward function R. The state set S contains every possible state $s_t \in S$ of the environment; the action set A contains every possible action $a_t \in A$ that the decision-making agent can exert to the environment; the state-transition probability P describes the probability $P(s'|s,a) = Pr\{s'|s,a\}$ that the environment transits from a current state s to a future state s' upon receiving an action a; the reward function $r = \mathcal{R}(s,a)$ measures how well a predefined goal has been achieved by an action a that is exerted to the environment at the state s. A subscript t is added here to denote that the associated state and action occur at a discrete time step t. On the other hand, a decision-making agent is commonly characterised by its policy function, which observes a state of the environment s and deduces an action a adaptively. The policy function can be categorised into two types of policies, namely the deterministic policy $a = \mu(s)$ and the stochastic policy $\pi(a|s) = Pr\{a|s\}$. The difference between these two types of policies is that a deterministic policy outputs an action directly, whereas the stochastic policy outputs a probability distribution of actions and therefore requires additional sampling in order to obtain an action. The choice for the stochasticity of the policy is usually determined by the design of the decision-making agent and the aim of the application. In Chapter 6, a deterministic policy is adopted, whereas in Chapter 7 a stochastic policy is adopted; the discussion about the stochasticity of the adopted policy be found in Section 7.10.

A control problem is said to be a Markov decision process if the state-transition probability of this control problem satisfies the Markov property:

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \cdots, s_1, a_1)$$

This means that the state transition to a future state s_{t+1} in this control problem has a sole dependence on the current state s_t and the action that is exerted to the environment at the current step a_t. In other words, the past history of states and actions does not affect the transition from the current state to a future state. The Markov property can also be interpreted as that each state must contain information of necessity to allow the decision-making agent to infer an action that transition the environment to every possible future state, which includes favourable and unfavourable states with respect to the controlling process (Sutton and Barto, 2018). Therefore, the interactions between the environment and the decision-making agent can be written in the form of a series of four-element tuples:

$$(s_1, a_1, r_1, s_2), (s_2, a_2, r_2, s_3), \cdots, (s_n, a_n, r_n, s_{n+1})$$

The sum of all rewards in this series can thus be written as:

$$G \stackrel{\text{def}}{=} r_1 + r_2 + r_2 + \cdots + r_n \qquad (3-5)$$

Here, G is referred to as the 'undiscounted episodic reward' (in some literature, G is referred to as the 'expected reward'), where 'episodic' implies that the summation of the rewards starts from the reward r_1 that is granted to a decision-making agent in the first interaction between this decision-making agent and an environment, and ends with the reward r_t which is granted to a decision-making agent in the last interaction between the decision-making agent and the environment. Through careful design of the reward function $r = \mathcal{R}(s, a)$ that assesses each state transition $s_t \rightarrow s_{t+1}$ with respect to a predefined goal, maximising G can be a sufficient condition for achieving the predefined goal. This transforms the control problem into a convex function optimisation problem.

The bounding properties of equation 3-5 depends solely on the reward function $r = \mathcal{R}(s, a)$. However, in practice, the reward function is more than often constructed based on empirical studies (e.g., Section 6.5.1), and therefore it cannot be guaranteed that the reward function contains sufficient bounding properties (Sutton and Barto, 2018). Therefore, in order to ensure the converging property of the undiscounted episodic reward, an exponential weighting parameter $0 < \gamma < 1$ is usually included in equation 3-5, which yields:

$$R \stackrel{\text{def}}{=} r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \gamma^{n-1} r_t = \sum_{t=1}^{N=n} \gamma^{t-1} r_t \qquad (3-6)$$

This R will be referred to as the 'discounted episodic reward' and the γ is commonly referred to as the 'discount factor'. The inclusion of the discount factor ensures the converging property of the summation of the rewards. However, it also weighs every future reward based on their respective distance (i.e., step count) to the present time period, and therefore an action that is closer to the present becomes more important in the decision-making process, compared to that of an action that is in later time step. In practice, in order to enable long-term planning, a discount factor near unity is often adopted. In Section 6.5.2, a more in-depth discussion in regard to the discount factor and its impact on the decision-making process is given.

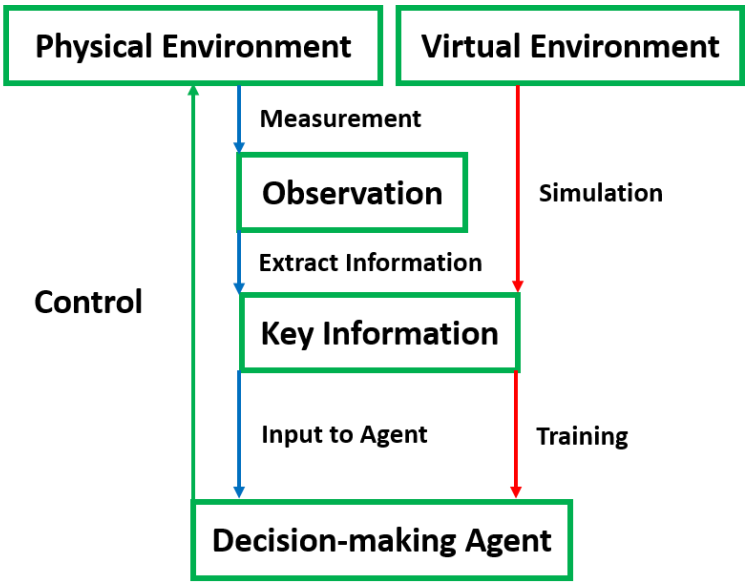

Figure 3-10 Schematic of training a decision-making agent in a virtual environment before application to a physical environment.

It is generally difficult to rigorous prove the Markov property of a real-world control problem, as the state set in a real-world problem tends to be very large. In this thesis, it is assumed that the Markov property holds true for the two demonstrated experiments in Chapter 6 and Chapter 7. These assumptions are then verified via their respective training, where only the current state is provided to the decision-making agent to deduce an action. As previously stated, the Markov property ensures that the state contains necessary information for a decision-making agent to deduce an action that transits this state to a desirable future state. This means that, in a real-world application where data collection is prohibitively expensive, it is possible to train a decision-making agent with simulated key information of the control process in a 'virtual environment' that approximates the associate physical environment. After training in the virtual environment, the trained agent can be then employed in the physical environment as a controller, where the trained agent uses the key information extracted from the observation of the control process as input, as presented in Figure 3-10. Similar concept has been proposed in the study of robotics (Zhao et al., 2020).

3.5 Reinforcement Learning

The studies of reinforcement learning can be summarised as the search for a policy function that is able to obtain the maximum discounted episodic reward (Sutton and Barto, 2018), given a Markov decision process. The reinforcement learning can be regarded as an extension of dynamic programming, where the core concept originates from Bellman's principle of optimality (Bellman, 1966):

'An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.'

The dynamic programming usually computes the value function $V_\pi(s)$ and the action-value function $Q_\pi(s,a)$ for a policy π:

$$V_\pi(s_t) = \mathbb{E}_\pi[R_t|s=s_t] = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r_{t+1} + \gamma V(s')] \quad (3-7)$$

$$Q_\pi(s_t,a_t) = \mathbb{E}[r_{t+1} + \gamma V_\pi(s')|s=s_t, a=a_t] = \sum_{s'} P(s'|s,a)[r_{t+1} + \gamma V(s')] \quad (3-8)$$

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{n=0}^{\infty} \gamma^n r_{n+t+1}$$

where $V_\pi(s)$ is the expectation of the weighted average of the discounted episodic reward at the state s following the policy π, and $Q_\pi(s,a)$ is the expectation of the discounted episodic reward at the state s upon exerting the action a following the policy π. In order to search for an optimum policy π^* (i.e., a policy function that is able to obtain the maximum discounted episodic reward), the value function for every state is estimated through averaging the obtained discounted episodic reward at the corresponding state following the policy π (this process is commonly referred to as the policy evaluation), and a new and improved policy π' can be constructed for every state via $\pi' = \underset{a}{\operatorname{argmax}}\, Q_\pi(s,a)$, in the sense that $\pi'(s) \geq \pi(s)$ holds true for every s, according to the policy improve theorem (Sutton and Barto, 2018) (this process is commonly referred to as the policy improvement). This process is repeated until the policy can no longer be improved, and an optimum policy π^* is therefore found:

$$\pi_0(s) \to V_{\pi_0}(s) \to \pi_1(s) \to V_{\pi_1}(s) \to \cdots \to \pi^*(s) \to V_{\pi^*}(s), s \in S \quad (3-9)$$

This control scheme is only applicable to control problems with small state sets, as both the policy evaluation and the policy improvement must be performed on every state in each iteration. This is sometimes referred to as the 'tabular method', which means that the space set is small and consequently can be kept track of with an array. In order to enable effective learning in real-world control problems that have relatively large state space, function approximators (e.g., neural networks) are usually used to represent the value function $V(s)$ and the action-value function $Q(s,a)$, and can be updated in an asymptotic and online manner to eventually converge (Sutton, 1988) (thus leading to an optimum policy):

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(S)] \quad (3-10)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma Q[s', \pi(a|s')] - Q(s,a)\} \quad (3-11)$$

where α is hyperparameter that controls the step size per update, and therefore, according to the naming convention in machine learning, is commonly referred to as the learning rate.

In the context of this thesis, the control problems that are of interest in Chapter 6 and Chapter 7 all possess a continuous action space. In other words, the control problems of interest require its action inputs a to be in a continuous domain (i.e., $\{a|m \leq a \leq n, a \in \mathbb{R}^t\}$ where m and n are the upper and lower limit of the interval and t is size of the action space). Note here that the stochasticity and continuity of an action space are two distinctively different concepts, where the former describes the parameterisation of the action, and the latter describes the categorisation of the action. In order to enable a continuous action space, the actor-critic method (Konda and Tsitsiklis, 1999) is adopted, where a function approximator represents the policy π. Here, the actor is referring to the policy π while the critic is commonly either the value function or the action-value function. The actor-critic method relays on the policy gradient theorem (Sutton et al., 1999) to find an optimum policy:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{\partial V(s)}{\partial \theta} \propto \sum_s d_\pi(s) \sum_a Q_\pi(s,a) \frac{\partial \pi(a|s;\theta)}{\partial \theta} \quad (3-12)$$

where $d_\pi(s) = \lim_{n\to\infty} Pr\{s|\pi\}$ is the probability distribution of the state s occurring under the policy π that is parameterised by θ. Equation 3-12 suggests that the parameters in the policy (i.e., actor) can be optimised towards the direction of maximising the value function, given a state s, and therefore the policy function can be optimised towards the direction of a local maximum.

Two reinforcement learning algorithms were used in the experiments presented in this thesis, namely the Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) and the Proximal Policy Optimization (PPO) (Schulman et al., 2017). The TD3 was used for the experiments in Chapter 6 while the PPO was used for the experiments in Chapter 7.

3.5.1 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Firstly, let ω denote the parameters of a deterministic policy function $\mu(s;\omega)$, also known as the actor, and θ denotes the parameters of an action-value function $Q(s,a;\theta)$, also known as the critic. The deterministic policy gradient theorem (Silver et al., 2014) gives:

$$\frac{\partial J(\omega)}{\partial \omega} = \mathbb{E}_{s\sim d_\pi(s)}\left[\frac{\partial Q_\mu(s,\mu(s;\omega);\theta)}{\partial \mu(s;\omega)} \frac{\partial \mu(s;\omega)}{\partial \omega}\right] \quad (3-13)$$

where the subscript of the expectation $s{\sim}d_\pi(s)$ means that the expectation is weighted by the probability distribution $d_\pi(s)$. This update direction is first employed in conjunction with neural networks acting as function approximators in the Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015). The TD3 is an improved algorithm over the DDPG, and the improvements are as follows (Fujimoto et al., 2018):

1) Double-Q learning to reduce accumulated error $Q_\mu(s,a;\theta) = \min_{\mathrm{i}=1,2} Q(s,\mu(s;\omega);\theta_i)$ (Van Hasselt et al., 2016).
2) Target policy smoothing to reduce variance, in equation 3-11, replacing π with $\mu_{smooth}(s') = \mathrm{clip}(\mu(s';\omega) + \mathrm{clip}(\epsilon,-c,c),a^-,a^+), \epsilon{\sim}\mathcal{N}(0,\sigma)$, where σ, c, a^- and a^+ are all hyperparameters that controls the smoothing.
3) Asymmetric policy evaluation and policy improvement, in pseudocode 3-9, performing multiple policy iteration before a policy improvement.

Note in equation 3-13, the gradient is an expectation that is weighted by a probability distribution $d_\pi(s)$. This distribution is normally inaccessible since it requires sampling over the entire state set. As a workaround, a 'replay buffer' (sometimes referred as the 'memory') is used to keep track of experienced tuples (s_t, a_t, r_t, s_{t+1}), and this probability distribution can be estimated through sampling from this replay buffer.

3.5.2 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a follow-up work based on Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), both of which are types of Minorize-Maximization (MM) algorithms. MM algorithms construct a surrogate function that is easy to optimise and represents a lower bound of a concave objective function, and through iteratively maximising the surrogate function, the objective function is driven uphill and eventually converges to a local or a global maximum. The 'surrogate' objective function of the TRPO is written as:

$$J(\omega) = \widehat{\mathbb{E}}_t\left[\frac{\pi(a|s;\omega)}{\pi(a|s;\omega_{\mathrm{old}})}\widehat{A}_\pi(s,a)\right] \; s.t.\, \widehat{\mathbb{E}}_t\{\mathrm{KL}[\pi(a|s;\omega_{\mathrm{old}}),\pi(a|s;\omega)]\} \le \delta$$

Where $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) = r + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)$ is the advantage function, which measures the extra cumulative reward that an agent obtains by choosing action a at state s_t compared to the average cumulative reward obtained at state s_t, KL is the Kullback–Leibler divergence that measures the numerical similarity between a probability distribution and a reference probability distribution, and δ is a hyperparameter that imposes a *trust region constraint* on the policy update (Schulman et al., 2015), which bounds the amount of change between an old policy and a new policy in the policy improvement process.

In order to increase the computational efficiency, TRPO approximates the 'surrogate' objective function by taking the Taylor series of the objective up to the first order, and the constraint up to the second order. To avoid the second order approximation and enable optimisation via the gradient ascent, the PPO simplifies the objective function to (Schulman et al., 2017):

$$J(\omega) = \widehat{\mathbb{E}}_t \left\{ \min \left[\frac{\pi(a|s;\omega)}{\pi(a|s;\omega_{\text{old}})} \hat{A}_\pi(s,a), \text{clip}\left(\frac{\pi(a|s;\omega)}{\pi(a|s;\omega_{\text{old}})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_\pi \right] \right\}$$

where a hyperparameter ϵ is used to clip the probability ratio (i.e., $\frac{\pi(a|s;\omega)}{\pi(a|s;\omega_{\text{old}})}$) when the difference between an old policy $\pi(a|s;\omega_{\text{old}})$ and a new policy $\pi(a|s;\omega)$ is exceedingly large (instead of using the Kullback–Leibler divergence).

3.6 Summary

Machine learning provides a diverse range of data-driven mathematical tools for a wide spectrum of applications. In particular, the artificial neural network possesses a tremendous ability to approximate other functions. In addition, multi-dimensional sensory signals can be effectively and efficiently processed by artificial neural networks with sparse connectivity, namely the convolutional neural network. Neural networks can be employed to map raw, real-world measurements of a process (e.g., camera observations) to interpretable descriptions about this process. Through training in a set of appropriate data, a neural network is able to learn the generality associated with the underlying process that is responsible for the generation of the training data. These qualities enable neural networks to represent fundamental functions (i.e., value functions, action-value functions and policy functions) that govern the optimum control in a Markov decision process. Therefore, the reinforcement learning agent that combines the neural network and dynamic programming allows real world control problems to be solved without any need for manual labour and human expertise.

Data is arguably the most important factor in training a generalised neural network. Therefore, data collection process must be carefully designed in order for correct training of the neural network. In the fields of light-matter interactions, data collection can be prohibitively expensive as it may requires tens of thousands of real-world samples to be processed by incident light. Data augmentation therefore provides an affordable means to expand a small dataset to a larger dataset, as discussed in detail in Section 5.4. Data collection is also an avoidable problem in real-world control problems. However, owing to the fact that the decision is inferred from the key information in the present observation of an environment in a Markov decision process, it is

possible to simulate the key information in a virtual environment for a reinforcement learning agent to master a certain skill. The agent can therefore be trained in a virtual environment before being employed in the corresponding physical environment.

All machine learning algorithms presented in this thesis are based on *Tensorflow* (Dillon et al., 2017, Abadi, 2016). The reinforcement learning agent presented in Chapter 6 is a modified version of the *Spinning Up* library (Achiam, 2018). The reinforcement learning agent presented in Chapter 7 is a modified version of *Stable Baselines* library (Hill, 2018).

Chapter 4 Microparticle Sensing in Solution via Convolutional Neural Network

A convolutional neural network can be used as a powerful data-driven function approximator, where the mapping from input to output can be learnt directly from experimental data. This capability enables the modelling of systems that are too time consuming or too complex to simulate directly from a theoretical understanding. Convolutional neural networks therefore have an important role to play in the modelling of complex light-matter interactions, such as where large numbers of interactions occur. In this chapter, it is shown that a convolutional neural network can be used to determine the properties of microparticles suspended in a liquid, directly from the scattering patterns recorded on a camera when the liquid is illuminated with a diode laser. The approach is shown to enable the determination of the microparticle concentration and type, even in mixtures of microparticles, whilst also identifying the salinity of the solution. This experiment was demonstrated using a Raspberry Pi that powered the diode source and the camera, and provided the computation for the convolutional neural network. This chapter therefore demonstrates an experimental setup that has the potential for deployment as a remote and low-cost sensor.

For the work in this chapter, the author created, optimised and trained the neural networks (including optimisation of architecture to be used on the Raspberry Pi), collected the experimental data, processed the experimental data to create the training and testing data, and performed all analysis. The author completed all challenges associated with running the computational component of the experiment, including optimisation of the Raspberry Pi camera, diode, and Wi-Fi. The experimental setup was jointly constructed by the author and Dr James Grant-Jacob.

4.1 Motivation

A single microparticle, or an ensemble of microparticles, will scatter incident light, where the angular distribution of the scattered light will depend on many factors (Bohren and Huffman, 2008). The scattered light can be recorded by various means, such as through imaging onto a camera via a microscopic objective and a tube lens. In general, the interaction of light with a spherical particle is modelled using Mie scattering theory, as it can offer a theoretically exact solution under certain conditions. In Mie scattering theory, the angular distribution of scattered light depends strongly on the wavelength and polarisation of the incident light, and the dielectric properties of the microparticle and the medium that contains the microparticle. Therefore, the scattered light contains information regarding the laser and the material, which unlocks the potential for

quantification of a particle directly from information recorded in a scattering pattern. When recording a scattering pattern, there is generally regarded as two domains, namely the near-field and the far-field, where the difference is determined by the relationship between the size of the particle, the wavelength of incident light, and the distance between the particle and the imaging plane (as defined by the Fresnel number).

In this work, the objective was to create a monitoring system based on a compact single-board computer (i.e., a Raspberry Pi) to determine, via a convolutional neural network, the following attributes of microparticles in suspension in deionised water or saline solutions via the camera observations of their scattering patterns:

1) The concentrations of the microparticles in the deionised water.
2) The salinities of the microparticle-contaminated saline solutions.

It will be later shown in this chapter that, despite being compact in size, the devised system was capable of determining these attributes precisely and accurately. This devised system could, with subsequent product development, potentially to be deployed in a marine environment, in order to monitor the salinity and the concentrations of contaminating microparticles of the marine environment. Microparticle contamination in marine environment has become an increasing health concern around the globe, in the sense that microbeads and microparticles that arise from human wastes and manufacturing processes can be accumulated in the organs of marine lives, which can be eventually consumed by humans (Bouwmeester et al., 2015).

The Mie theory, as well as other well-established theories that concern the scattering of light, provide the necessary mathematical tools for simulating the scattering patterns from a range of objects. Specifically in Mie theory, the scattering pattern resulting from illumination of a homogenous sphere in both far field and near field can be simulated in an excellent agreement with experiment observations (see Section 2.3 for a general description of the Mie theory). However, the theoretical calculations can become very complicated in cases where 1) the object is not of a spherical shape (Hinojosa-Alvarado and Gutiérrez-Vega, 2010), 2) the object is of multiple separable parts (e.g., an ensemble of microparticles (Gouesbet and Gréhan, 1999)), 3) the object is inhomogeneous (Hightower et al., 1988), or 4) the object is embedded in a medium (Frisvad et al., 2007). In practice, it is possible for multiple factors to simultaneously affect the scattering pattern from an object. Therefore, although there exist other similar studies that utilise scattered patterns of microparticles in solutions to determine the attributes of the microparticles, these studies often limit their subject microparticles to ideal scenarios (Philips et al., 2017), for example limiting the number of microparticles that are radiated by incident light, in order to simplify the associated theoretical calculations. However, by employing a convolutional neural network as a function

approximator, the scattering patterns of microparticles in a non-ideal scenario (e.g., aggregated inhomogeneous microparticles in translucent solutions) can be investigated and analysed, as the experimental data itself can be used to train the network. As a result, the numerical values that describe the attributes of these microparticles (e.g., the concentration of the microparticles and the salinities of the solution), can be identified directly from the experimentally recorded scattering patterns, even for samples that would be far too complex to model using theoretical approaches.

4.2 Experimental Setup

The schematic of the experimental setup is depicted in Figure 4-1. Laser from a 1 mW, 650 nm diode laser (*Thorlabs* L650P007) was focused via a lens (f = 2.5 cm) into a transparent sample holder that contained microparticles in suspension in either deionised water or a saline solution. The focused spot size of the laser inside the sample holder was approximately of an elliptical shape, with major and minor axis of ~20 µm and ~10 µm, respectively. The light scattered from the irradiated microparticles in the sample holder was projected onto a piece of white polyester screen that was positioned 1.5 cm after the sample holder and was orthogonal to the incident laser beam. A camera (*Raspberry Pi* Camera Module CSI-2, resolution of 2464 × 3280) was placed behind this screen to image the scattering pattern from the irradiated microparticles in real time. This camera was connected to a compact single-board computer (*Raspberry Pi* 3 Model B+) that hosted a convolutional neural network for determining the concentration of the microparticles, and the salinity of the solution, based on the camera observation of the scattered pattern.

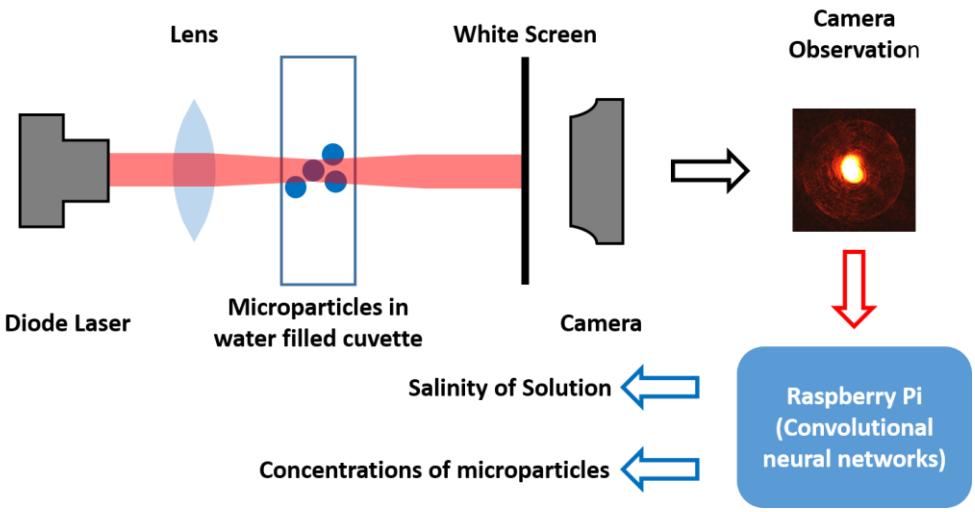

Figure 4-1 Schematic of experiment setup.

The sample holder was a transparent cuvette, with exterior dimensions of 12.5 × 12.5 × 45 mm. Two types of microparticles were used in this study, namely silicon dioxide microspheres (5 µm ± 100 nm in diameter, Sigma Aldrich 44054-5ML-F) and melamine resin

microspheres (8 µm ± 200 nm in diameter, Sigma Aldrich 95523-5ML-F). The interior dimension of the cuvette along the direction of the incident beam was approximated 10 mm, which in turn provided an approximate volume of 10 µm × 5 µm × π × 10 mm ≈ 1,570,750.0 μm^3 for light-matter interactions (i.e., light scattering) to occur. Considering that the volumes of a single silicon dioxide microsphere and a single melamine resin microsphere were approximated 65.5 μm^3 and 268.1 μm^3 respectively, a maximum number of 23,980.9 silicon dioxide microspheres and a maximum number of 5858.8 melamine resin microspheres could simultaneously interact with a beam of light in this sample holder (without considering close-packing). The saline solution was prepared through dissolving sodium chloride in deionised water, and the microparticles were mixed to the saline solution prior to the start of each experiment via a pipette. Both the concentrations of the microparticles and the salinities of the solutions are described in parts per thousand by mass (ppt). An example of the scattered light from a prepared sample that contains a mixture of 0.050 ppt silicon dioxide microparticles and 0.050 ppt melamine resin microparticles in deionised water is shown in Figure 4-2.

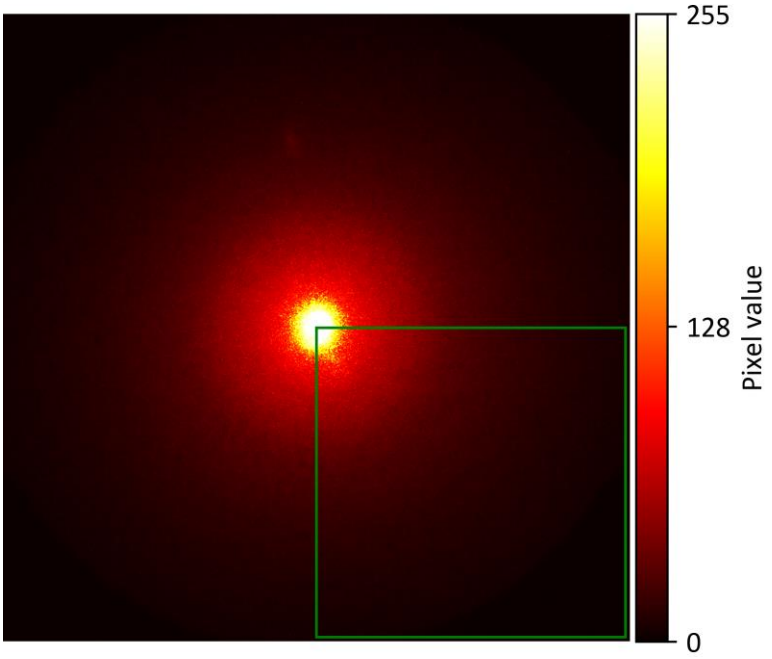

Figure 4-2 Camera observation of the scattering pattern from a sample solution with silicon dioxide microspheres (0.050 ppt) and melamine resin microspheres (0.050 ppt) in deionised water. The green rectangle illustrates the cropped area of the lower right quadrant of this camera observation, which was used as the input to the convolutional neural network.

The same setup was also used for collecting the training dataset, which was formed from the frames from the video recordings of the scattering patterns for each prepared sample. For efficiency, it was chosen that only the lower right quadrant of the scattered patterns was used as

inputs to the convolutional neural networks (shown in a green rectangle in Figure 4-2), as it was understood that each quadrant would contain the same (or very similar) information due to symmetry. The quadrants of the scattered patterns were resized to 100 × 100 pixels. Fresh samples (i.e., not the same samples used for collecting the training dataset) were prepared at the start of the testing experiments, in which the trained convolutional neural networks were used to determine the salinity or the concentration of the microparticles of the freshly made samples. Note that, as per the training, the lower right quadrant of the scattered patterns was used as the input to the convolutional neural networks during testing. The sample preparation, the training dataset collection and the testing experiments were all conduced at 22 degrees Celsius.

4.3 Neural Network Architecture

The neural network architecture that was used for the experiment in Section 4.4 is shown in Table 4-1. Owing to the fact that in Section 4.4 the neural network outputted two concentrations for two types of microparticles simultaneously, the output of the final layer in this neural network was of the shape 1 × 2, which corresponds to the two determined concentrations of silicon dioxide microspheres and melamine resin microspheres respectively.

Layer name	Input size (pixels)	Parameter (pixels)	Output size (pixels)
Input layer	2464 × 3280 × 3	-	100 × 100 × 1
Conv_1	100 × 100 × 1	3 × 3 × 64	100 × 100 × 64
Maxpooling_1	100 × 100 × 64	2 × 2, stride of 2	50 × 50 × 64
Conv_2	50 × 50 × 64	3 × 3 × 64	50 × 50 × 64
Maxpooling_2	50 × 50 × 64	2 × 2, stride of 2	25 × 25 × 64
Flatten	25 × 25 × 64	-	1 × 40000
Drop_out (Srivastava et al., 2014)	1 × 40000	-	1 × 40000
Fully_Connected	1 × 40000	40000 × 1024	1 × 1024
Output	1 × 1024	1024 × 2	1 × 2

Table 4-1 Neural network architecture.

The neural network architecture that was used for the experiment in Section 4.5 is shown in Table 4-2. Note that since this neural network was only tasked to determine the salinity of the microparticle-contaminated solution, the output of this neural network was of the shape 1 × 1. Other parameters can be found in Table A 1. In both tables, the output of the upper layer is the input of the lower layer.

Layer name	Input size (pixels)	Parameter (pixels)	Output size (pixels)
Input layer	100 × 100	-	100 × 100
Conv_1	100 × 100	3 × 3 × 64	100 × 100 × 64
Maxpooling_1	100 × 100 × 64	2 × 2, stride of 2	50 × 50 × 64
Conv_2	50 × 50 × 64	3 × 3 × 64	50 × 50 × 64
Maxpooling_2	50 × 50 × 64	2 × 2, stride of 2	25 × 25 × 64
Flatten	25 × 25 × 64	-	1 × 40000
Drop_out	1 × 40000	-	1 × 40000
Fully_Connected	1 × 40000	40000 × 1024	1 × 1024
Output	1 × 1024	1024 × 1	1 × 1

Table 4-2 Neural network architecture.

4.4 Identification of Microparticle Concentration

In the first demonstration, the convolutional neural network was employed to simultaneously determine the concentrations of both the silicon dioxide microparticles and the melamine resin microparticles, where both types of microparticles were present in the deionised water, directly from the recorded scattering patterns of the sample. In this case, the scattering patterns therefore contained information regarding the concentration of both types of particles. In order to collect the appropriate training dataset, nine samples that contained mixtures of both types of microparticles in deionised water were created. For all nine samples, the numerical values of the concentrations for the two types of microparticles added up to be 0.1000 ppt. In addition, the concentration for each type of the microparticle was incrementally increased from 0.0000 to 0.1000 ppt with an increment of 0.0125 ppt across all nine samples. This means that, for instance, if the concentration of the silicon dioxide microparticles was 0.0125 ppt in a sample, then the concentration of the melamine resin microparticles in the same sample was 0.1 ppt - 0.0125 ppt = 0.0875 ppt. The

combinations of the concentrations for both the silicon dioxide microparticles and the melamine resin microparticles for these nine samples are listed in Table 4-3. As per the description in Section 4.2, nine videos of the associated scattered patterns were recorded when each of these nine samples were installed in the experimental setup one at a time. Each video contained 100 frames of the associated scattering patterns of the microparticles, and therefore a total number of 900 frames (i.e., 900 images) were collected as training materials, where 90 % of these training materials were used to constitute the 'training dataset'. The scattering patterns and their corresponding labels (i.e., the numerical values of the concentrations in ppt for both the silicon dioxide microparticles and the melamine resin microparticles) from this training set were repeatedly fed to the convolutional neural network during training, so that the convolutional neural network learnt to map camera images of scattering patterns to their respective labels (i.e., feed forward and back propagation, see Section 3.2). The remaining 10 % of the training materials were used to constitute a 'validation dataset'. The camera images from the validation dataset were fed to the convolutional neural network after one iteration of training, and the prediction accuracy was used to quantify a generalisation error (see Section 3.2). This generalisation error was the discrepancy between the accuracies of the convolutional neural network in determining the concentrations of the camera observations of microparticles from the training set and the validation set, and this error represented the degree of overfitting.

Sample number (#)	Concentration of the silicon dioxide microparticles (ppt)	Concentration of the silicon melamine resin (ppt)
0	0.0000	0.1000
1	0.0125	0.0875
2	0.0250	0.0750
3	0.0375	0.0625
4	0.0500	0.0500
5	0.0625	0.0375
6	0.0750	0.0250
7	0.0875	0.0125
8	0.1000	0.0000

Table 4-3 Combinations of concentrations of microparticles in each sample for training.

Once the accuracy of the convolutional neural network in determining the concentrations of microparticles for the training set plateaued at a satisfactory high level and the discrepancy

between the accuracies of the convolutional neural network for the training set and the validation set was at a satisfactory low level (i.e., indicating that the generalisation error is low), the training of the convolutional neural network was terminated. The trained convolutional neural network was then incorporated into the single-board computer, and was used in an experiment, where the convolutional neural network determined the concentrations of microparticles for freshly made samples in real time. Two separated testing experiments were conducted at two different dates, firstly one day after the date when the training was completed and secondly twenty days after the date when the training was completed. In the first part of the experiment, which was completed one day after training was completed, three samples were freshly prepared and then their scattering patterns were analysed by the trained convolutional neural network, each of which contained a mixture of the two types of microparticles whose concentrations are listed in Table 4-4. Note that the numerical values of these concentrations still added up to be 0.1000 ppt, but the combinations of the individual concentrates differed from that of the training materials. The second part of the experiment was conducted twenty days after training was completed, in which two samples were freshly prepared and tested by the convolutional neural network. The motivation for a 20-day delay before testing, was to evaluate the resilience of the trained neural network when exposed to small changes in the nature of the recorded scattering patterns. For this reason, the entire experimental setup was broken down to pieces and then reconstructed between the first and the second experiment.

Sample number (#)	Date of testing (days)	Concentration of the silicon dioxide microparticles (ppt)	Concentration of the melamine resin microparticles (ppt)
0	1	0.0100	0.0900
1	20	0.0200	0.0800
2	20	0.0400	0.0600
3	1	0.0375	0.0625
4	1	0.0900	0.0100

Table 4-4 Combinations of concentrations of microparticles in each sample for the experiment.

In each of the two testing experiments, 10 measurements of each sample were collected, with a time interval between two consecutive measurements of 10 ms. The results of the two experiments are shown in Figure 4-3, where the results that were obtained 1 day after training was completed is plotted in red, and the results obtained twenty days after training was completed is

plotted in green. It is evident from the figure that, regardless of the date by which the experiment was conducted, the convolutional neural network could accurately determine the concentrations of both the silicon dioxide microparticles and the melamine resin microparticles in deionised water. However, the convolutional neural network was indeed less accurate in the case for twenty days after the completion of training. A likely reason for this is that the experimental setup was deconstructed and then reconstructed prior to the second experiment, and therefore the beam alignment in the second experiment may not have been identical to that in the first experiment (and indeed during the collection of training data). Since only the lower right quadrant of the scattered patterns were observed by the convolutional neural network, it is possible that a slightly misaligned setup could lead to a slightly different centre of the scattering patterns, and therefore lead to a lower precision in identification of the concentrations of the microparticles. The time delay between the two parts of the experiment and the freshly prepared sample was made for a similar reason. A piece of information in an input image to a convolutional neural network can be hardware-specific, date-specific and sample-specific, and thus if this information also contributes to the mapping of the input image to certain attributes, this mapping is therefore constrained by the hardware, the date and the sample (i.e., a false positive or a false negative). The accuracy can be characterised via the coefficient of determination (denoted as R^2).

$$R^2 = 1 - \frac{\sum_{t=1}^{N}(Prediction_t - GroundTruth_t)^2}{\sum_{t=1}^{N}\left(Prediction_t - \frac{1}{N}\sum_{t=1}^{N}(Prediction_t)^2\right)^2} \qquad (4-1)$$

The R^2 for the silicon dioxide and the melamine resin measurements, including the data that were measured twenty days after the completion of training, were approximately 0.9902 and 0.9968 respectively.

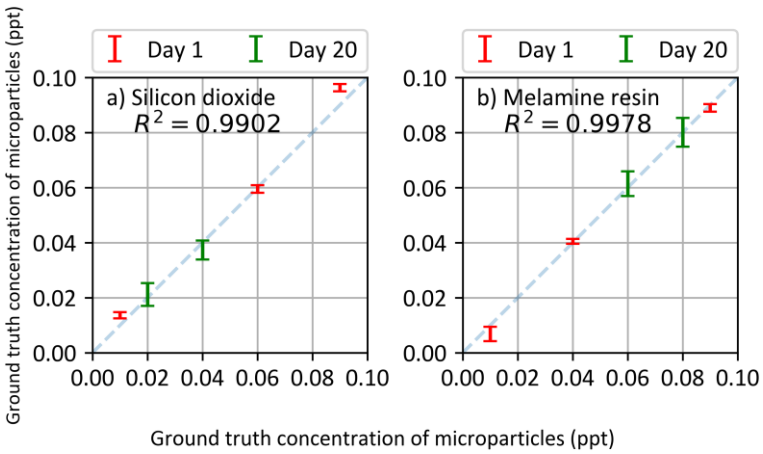

Figure 4-3 Accuracy of the trained convolutional neural network in the simultanous determination of concentrations of two types of particles when in a mixture.

4.5 Identification of Solution Salinity

In the second experiment, the convolutional neural network was tasked to determine the salinity of a range of microparticle-contaminated saline solutions. Similar to the data collection process that was previously described in Section 4.4, 11 samples were created, for recording 11 videos of their respective scattering patterns. All prepared samples contained a fixed concentration (0.1 ppt) of melamine resin microparticles and an increasing concentration of sodium chloride, starting from 0 to 100 ppt with an increment of 10 ppt. The combinations of components of all 11 samples are listed in Table 4-5. In each video, 100 frames of the camera observations of the scattering patterns were collected, and therefore a total number of $11 \times 100 = 1{,}100$ images from these 11 videos constituted the training materials. As per the discussion in Section 4.4, the training materials was split into the training set and the validation set with a ratio of 9-to-1, and the convolutional neural network was trained with the camera observations of the scattered patterns and their respective labels from the training set, while the generalisation error was similarly evaluated.

Sample number (#)	Concentration of the melamine resin microparticles (ppt)	Concentration of the Sodium chloride (ppt)
0	0.1	0.0
1	0.1	10.0
2	0.1	20.0
3	0.1	30.0
4	0.1	40.0
5	0.1	50.0
6	0.1	60.0
7	0.1	70.0
8	0.1	80.0
9	0.1	90.0
10	0.1	100.0

Table 4-5 Combinations of concentrations of components in each sample for training.

Once the accuracy of the convolutional neural network in determining the salinities for the training set plateaued at a satisfactory high level, and the generalisation error was at a satisfactory

low level, the training of the convolutional neural network was terminated. The convolutional neural network was then incorporated into the single-board computer, in order to determine the salinities of freshly prepared samples, directly from their respective camera observations of scattered patterns. A total number of seven samples were freshly prepared prior to the testing experiment, each of which contained a fixed concentration of 0.1 ppt melamine resin microparticles, and with one of a set of different concentration of sodium chloride. These different concentrations of sodium chloride were 0, 3, 8, 18, 35, 65 and 85 ppt, which corresponded to the concentrations of sodium chloride of fresh water, agriculture irrigation, Baltic sea, Black sea, average sea water, Hamelin pool and Lake Urmia, respectively. A total number of ten measurements were taken for each sample with a 10 ms time interval between consecutive measurements, and the results are presented in Figure 4-4. The R^2 for the salinity measurements was 0.999, which confirms that the convolutional neural network was able to accurately determine that salinities of the freshly prepared samples.

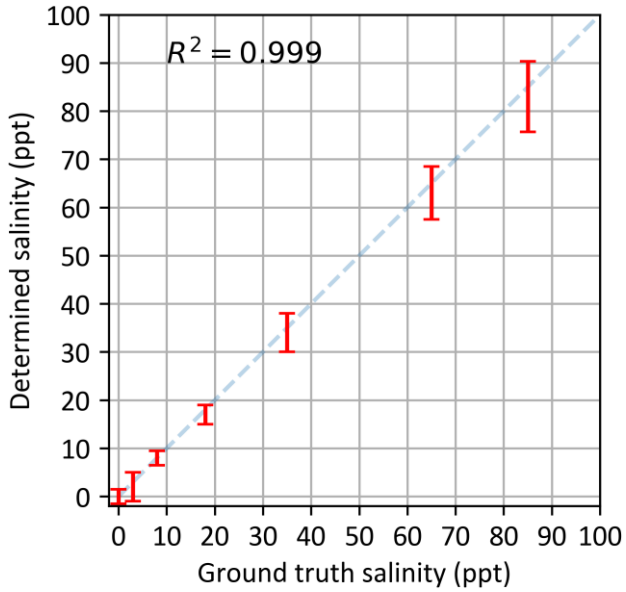

Figure 4-4 Accuracy of the trained convolutional neural network in the determination of silnities of microparticle-contaminated saline solutions.

4.6 Conclusions

In conclusion, two applications were demonstrated that utilise the convolutional neural network for the determination of attributes of microparticles when are in suspension in a solution, directly from the scattering pattern resulting from illumination of the solution. The first application utilised the convolutional neural network to simultaneously determine the concentrations of two types of microparticles, based on the camera observations of the scattering patterns of the mixture solution. The second application utilised the convolutional neural network to determine the salinity of a

microparticle-contaminated saline solution, based on the camera observations of the scattering patterns from the saline solution. The convolutional neural networks were able to determine these attributes through taking multiple measurements with an accuracy in excess of $R^2 = 0.99$. In addition, the convolutional neural networks were hosted by a compact Raspberry Pi single-board computer that was integrated into a compact system, which could possibly be used in marine environments for monitoring contaminating microparticles and salinity of the marine environments.

Chapter 5 Convolutional Neural Networks for Monitoring and Controlling Laser Machining

Femtosecond laser machining is a well-established manufacturing technique for high precision non-contact material removal. However, instabilities in laser parameters (and indeed material inhomogeneities) can lead to incorrectly machined samples. There is therefore great interest in techniques for real-time control of precision laser machining. In this chapter, a convolutional neural network is applied for the automated identification of beam translation and rotation during laser machining, via observation of the machined sample. Such an approach could be used to form the observation component of a real-time monitoring system. Following this, a second convolutional neural network was used to cease laser machining at the point where the top layer of a multilayer structure was machined, despite the neural network not having information regarding the thickness of the top layer. This was achieved via integration of the neural network in a real-time feedback loop with a camera that observed the sample during machining and the laser pulse controller.

For the work in this chapter, the author created, optimised and trained the neural networks, collected the experimental data, processed the experimental data to create the training and testing data, and performed all analysis. The author proposed the data augmentation approach to reduce the reliance on the data collection. The experiment was conducted by the author with the help of Dr Daniel Heath.

5.1 Motivation

In high precision laser materials processing, a major concern is that variations and instabilities in the associated experimental apparatus could introduce inadvertent modifications to the intended processes of laser-matter interactions. These variations and instabilities include, for example, the instability in the output power of a laser source, or the vibrations and the temperature/humidity changes that cause beam delivery optics to behave differently than expected. The resultant modifications comprise, for instance, in variations in the positions at which laser pulses are incident, or distortions to the spatial intensity profiles of an incident pulse. Extra precautions are usually factored in to avoid these modifications, such as building the optical system on a vibration isolation system (e.g., optical benches), or carefully maintaining the temperature and the humidity in the environment where the optical system resides. As generally,

1) The spatial intensity profiles of laser pulses.
2) The fluence of laser pulses.

3) The positions on a workpiece where laser-matter interactions occur.

determine the final quality of a laser machined target sample, there is increasing awareness of the need for real-time monitoring systems that can observe (and indeed correct for) the laser machining process. In some cases, due to potential instabilities in laser parameters and beam delivery optics, the exposure time or output power of a laser source will be deliberately tuned to exceed the theoretical optimal value, for ensuring that the underlying laser materials processing can be completed with a high certainty. A more efficient and effective solution would therefore involve a monitoring system that is capable of collecting necessary information (i.e., visual and metric properties) about the laser-induced materials processing in order for a controller (e.g., PID controller) to determine the best courses of action for each point in time throughout the laser machining process.

In this chapter, a monitoring system that incorporates convolutional neural networks was firstly demonstrated for detecting the positional offset and the rotational angle of a femtosecond laser machining process with an elliptically-shaped spatial intensity profile. This detection was achieved by a convolutional neural network that processed the camera images of the target sample. This proof-of-principle demonstration illustrated that a convolutional neural network could be used to monitor, in real time, angular and positional modifications to the spatial intensity profile of laser pulses that are inadvertently applied during laser machining, through observing the camera observation of the laser-machined structure. A second experiment was conducted in the same system, where femtosecond laser pulses were applied to the same position on a thin-film-deposited workpiece. A convolutional neural network was used to determine the moment at which the laser pulses should be halted, in order for the thin film to be removed (via laser machining) without also machining the substrate material below the thin film.

5.2 Experimental Setup

The experiment setup is schematically depicted in Figure 5-1. In this setup, laser pulses (1 mJ, 150 fs at 800 nm, *Coherent* Mira Optima 900-F) illuminated the central region of a DMD (*Texas Instruments DLPC3000*). See Section 2.2 for details describing the mechanism and application of the DMD. The DMD here was used to artificially apply controllable modifications to the spatial intensity profile of incident pulses. The modified pulses were then imaged onto a target workpiece, which was stationed on a sample holder, via a microscope objective (*Nikon ELWD 50*×). This sampler holder was supported by a set of XYZ translation stages, where the XY stages were set orthogonal to each other and to the direction of incident pulses, and the Z stage was aligned along the direction of the incident pulses. In the experiments, the XY stages were used exclusively for moving the

workpiece to a fresh area (i.e., area without any previously laser-machined structures) at the start of the experiments, while the Z stage was used in the experiments to adjust the vertical position of the target workpiece in order for the front surface of the target workpiece to be maintained at the beam focus (via an auto-focusing algorithm that measures the spatial frequencies of the observable area on the workpiece) before each incident pulses. A dichroic mirror was positioned above the microscope objective, allowing for simultaneous imaging of the target workpiece through a CMOS camera (*Thorlabs DCC1545M*). The workpiece target sample used in the first experiment was a 5 µm-thick electroless-nickel deposited on copper, and the fluence of the incident pulses was ~1.22 J/cm^2. Other studies have suggested that this is a viable fluence to machine nickel via the ablation effect (Willis and Xu, 2002).

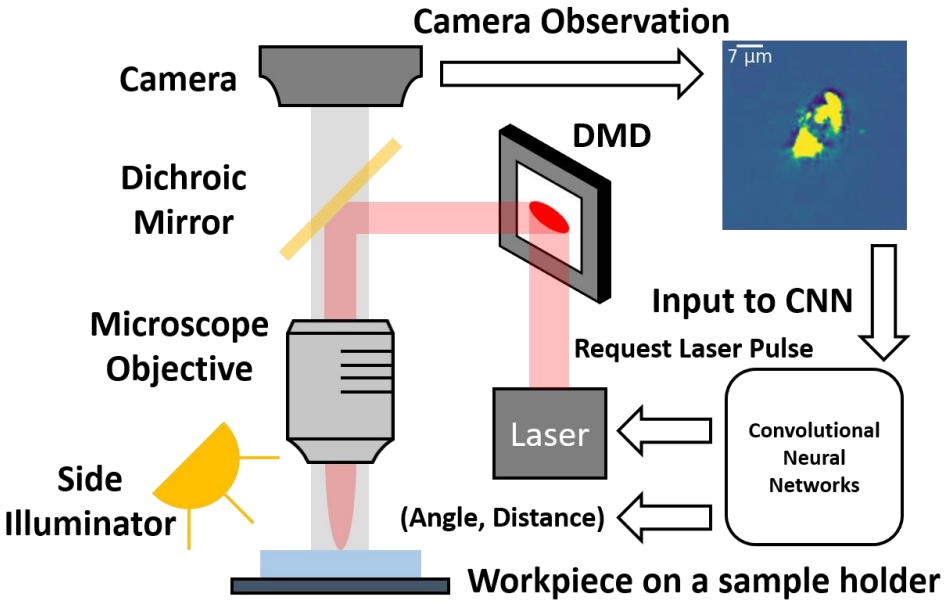

Figure 5-1 Schematic of the experimental setup.

In the first experiment, the DMD was utilised as a binary spatial light modulator, which modified the spatial intensity profile of the incident laser pulses to have a spatial shape of an ellipse with a uniform intensity. The uniform intensity was achieved through the use of a 'top-hat' beam shaper (*AdlOptica πShaper6_6*)) used before the DMD. The micro-mirror array on the DMD was used to display an elliptical shape, and hence the DMD functioned as an elliptically-shaped binary intensity mask. In addition, both,

1) The rotation angle of the major axis of the elliptical light mask about the X-axis of the DMD.
2) The distance between the centre of the elliptical light mask and the centre of the DMD.

were independently and simultaneously adjustable, as illustrated in Figure 5-2. The imaged laser pulses after the elliptical light mask were used to machine elliptical structures on the target workpiece via the laser ablation. The lengths of the major and minor axes of the elliptical laser-

machined structures were 14 µm and 7 µm respectively, where application of the two adjustable modifications (i.e., position and angle) corresponded to associated changes in the position and angle of the laser machined structure. Specifically, rotating the elliptical light mask at θ (about the X-axis of the DMD) yielded approximately a rotation θ of the elliptical laser-machined structures. Additionally, moving the elliptical light mask by one micro-sized mirror along a DMD axis (X- or Y-axis on the DMD) would result in the translation of the elliptical laser-machined structure along a corresponding major or minor axis by 91 ± 15 nm. The convolutional neural networks were used in this experiment to determine both 1) the angle of rotation, characterised by θ, and 2) the distance between the centre of the elliptical light mask and the centre of the DMD micro-mirror array, characterised by (d_x, d_y), where the only information provided was the camera observation of the laser machined structure.

Whilst other methods exist for detecting the translation and rotation of a shape with respect to a reference (e.g., phase-correlation algorithm (Zitova and Flusser, 2003) and Kalman filter (Li et al., 2015)), the convolutional neural network offers a generalised solution for monitoring this laser machining processes. Rather than requiring a set of programmatical descriptions for determination of the translation and rotation, here the convolutional neural network achieves this via training on experimental data, hence highlighting the potential for this technique to be used for identification of more complex aberrations.

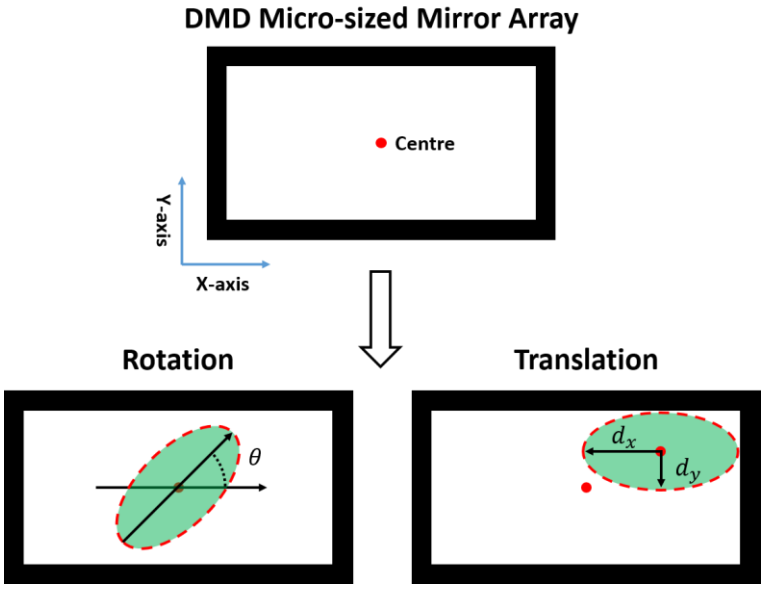

Figure 5-2 Schematic of the adjustable modifications, namely the rotation, which is characterised by θ, and the translation, which is characterised by (d_x, d_y), of the elliptical light mask.

Using the same experimental setup, a convolutional neural network was applied to determine the number of incident pulses that were still required to remove the entirety of a thin

layer of materials that was deposited on a substrate, where the objective was to machine the top layer completely with zero damage to the underlying substrate. In this experiment, artificial modifications were not applied to the elliptical light mask on the DMD (i.e., the spatial intensity profile was constant in terms of position and angle), but the elliptical light mask was nevertheless used in order to maintain consistency with the previous experiment. The elliptic light mask therefore had a fixed rotation angle $\theta = 0$ and a fixed translation distance $(d_x = 0, d_y = 0)$. In practice, other top-hat spatial intensity profiles could have also been similarly used. A real-time feedback loop was built, which incorporated the convolutional neural network as a controller that governed, at a discrete time step, whether a laser pulse should be applied to the workpiece at the current time step given the current camera observation of the workpiece. The depiction of the flow diagram of this process is shown in Figure 5-1, and it worked as follows:

1) A camera observation of the workpiece was captured prior to the incident pulse in the current step.
2) This observation was sent to the convolutional neural network as an input, and the convolutional neural network outputted a value that determined the remaining number of incident pulses which were still required for only penetrating the thin film, which was deposited on the substrate.
3) If the returned value was positive, which determined that further incident pulses were still required, a single additional pulse would be applied at the thin film in the current step; if the returned value was a non-positive (i.e., zero or a negative value), which determined that the thin film had been penetrated, pulse irradiation would be halted immediately, and the XY translation stages would move the sample to a fresh area.

The incident pulses to the same position on the workpiece was only halted under the condition that the convolutional neural network outputted a non-positive value. An illustration of the concept of this controlling process is depicted in Figure 5-3. The thin-film-deposited workpiece that was used in this experiment was a ~450 nm thick sputtered copper layer on silica, and the fluence of the incident pulses was ~1.83 J/cm^2, which was above the ablation threshold for copper (Jandeleit et al., 1996).

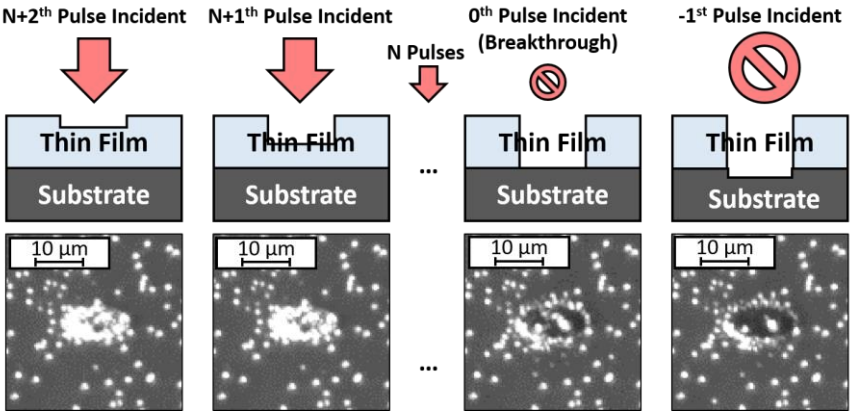

Figure 5-3 Schematic of the controlling process, where the red arrow indicates that a pulse will be applied to the workpiece. Here the state of the workpiece in which the thin fim has been penetrated but the substrated has not been machined is referred to as the 'breakthrough' points.

In order to ensure that the thin film was machined through, the surface profile of the workpiece was exanimated via a Scanning White-Light Interferometry (SWLI, *zygo ZeScope*). This profilometer measures the interference between a beam that is reflected or scattered from a surface and a reference beam, to determine the height of the surface. An example of the surface profile of an elliptical laser-machined structure, which was measured via the ZeScope profilometer, is shown in Figure 5-4. This elliptical laser-machined structure was machined with 50 consecutive incident pulses, and the depth of machining is approximately 502 ± 165 nm.

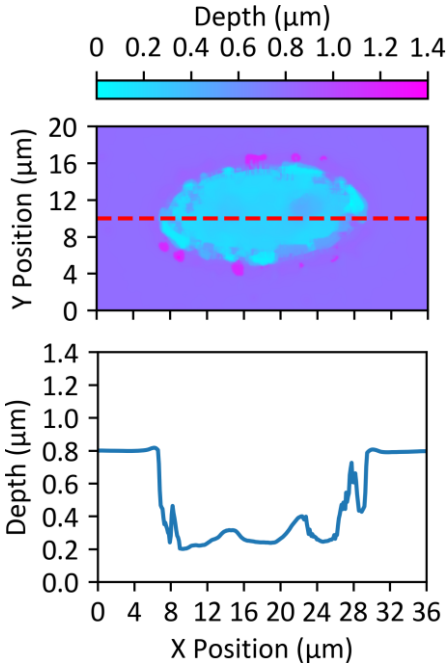

Figure 5-4 Surface profile of an elliptical laser-machined structure. The lower figure shows the surface profile of the elliptical laser-machined structure along the dash red line shown in the upper figure.

5.3 Neural Network Architecture

Table 5-1 presents the connectivity of the neural network that was used in the experiments to determine the translations of the elliptical laser-machined structures in Section 5.5 and Section 5.6, and to determine the number of pulses to penetrate a thin film is Section 5.7. In these experiments, the camera observations of the elliptical laser-machined structures were mapped to a single value, and therefore the output of the neural network was of the shape 1 × 1.

Layer name	Input size (pixels)	Parameter (pixels)	Output size (pixels)
Input layer	100 × 100 × 3	-	100 × 100 × 1
Conv_1	100 × 100 × 1	5 × 5 × 20	100 × 100 × 20
Maxpooling_1	100 × 100 × 20	2 × 2, stride of 2	50 × 50 × 20
Conv_2	50 × 50 × 20	5 × 5 × 20	50 × 50 × 20
Maxpooling_2	50 × 50 × 20	2 × 2, stride of 2	25 × 25 × 20
Flatten	25 × 25 × 20	-	1 × 12500
Fully_Connected	1 × 12500	12500 × 1024	1 × 1024
Output	1 × 1024	1024 × 1	1 × 1

Table 5-1 Neural network architecture.

The neural network that was responsible for determining the rotation angle in Section 5.6 is shown in Table 5-2. Owing to the reasons which will be discussed in Section 5.6, this neural network determined the sine and cosine of the rotation angle simultaneously, which in turn causes the final output to possess the shape of 1 × 2. Other parameters can be found in Table A 2. In both tables, the output of the upper layer is the input of the lower layer.

Layer name	Input size (pixels)	Parameter (pixels)	Output size (pixels)
Input layer	100 × 100	-	100 × 100
Conv_1	100 × 100	3 × 3 × 64	100 × 100 × 64
Maxpooling_1	100 × 100 × 64	2 × 2, stride of 2	50 × 50 × 64
Conv_2	50 × 50 × 64	3 × 3 × 64	50 × 50 × 64

Maxpooling_2	50 × 50 × 64	2 × 2, stride of 2	25 × 25 × 64
Flatten	25 × 25 × 64	-	1 × 40000
Fully_Connected	1 × 40000	40000 × 1024	1 × 1024
Output	1 × 1024	1024 × 2	1 × 2

Table 5-2 Neural network architecture.

5.4 Training Data and Augmentation

Owing to the data-driven nature of the convolutional neural network, it is generally assumed that a significant amount of training data is needed. Whilst there exist studies that focus on investigating the data efficiency of the convolutional neural network (Du et al., 2018), in general, key breakthrough results in the literature indicate that successful training of a convolutional neural network with a deep architecture requires millions of training data items. For instance, the standardised testbed for image classification, ImageNet (the Large Scale Visual Recognition Challenge (Krizhevsky et al., 2012)), contains 1,281,167 labelled images as the training material. To this end, the collection of the suitable training materials is one of the most, if not the most, time-consuming and challenging processes in obtaining a high performing convolutional neural network. Of course, in the general case of laser materials processing, where each training data item may correspond to the physical machining of a single sample, it is therefore even more time-consuming and challenging to collect a satisfactory number of training examples. If one assumes that a single data item can be collected every second, the collection of 1 million data items would take approximately twelve days. Clearly, this would not be practical. This is a common concern which is encountered in other scientific communities, and therefore workarounds have been studied to mitigate the dependency of the convolutional neural network (or more generally, any data-driven deep learning architectures) on the number of training materials. One proposed method is ‘data augmentation’, in which a smaller set of training materials can be expanded to produce a set of training materials that is of a larger size. There are a wide range of image augmentation techniques, such as stretching, cropping, translating, rotation and even changing the brightness and contrast (Shorten and Khoshgoftaar, 2019). For the first experiment demonstrated in this chapter, data augmentation was used to crop images corresponding to the camera observations of an elliptically shaped laser-machined structure at different positions, in order to artificially introduce a translation to the position of the elliptical laser-machined structure. This was possible to achieve because the elliptical laser-machined structure only occupied a small fraction of the camera image of the workpiece — the CMOS camera which collected observations of the workpiece had a resolution of 1280 × 1024 pixels, and one elliptical laser-machined structure had an approximate size of 50 × 50

pixels on the observation. Figure 5-5 visually depicts this data augmentation process, where one single camera observation of a laser-machined structure is cropped at three different positions. These three instances of cropping result in three separate images, where each image shows an elliptical laser-machined structure that has a different translation between the centre of the ellipse and the centre of the image. The resultant images (after cropping) are therefore visually similar to the observations of the elliptical laser-machined structures that are machined with the modified elliptical light masks (i.e., the elliptical light masks that is translated to three different (d_x, d_y),) and hence can be used to train a convolutional neural network to detect the translation of the beam position during laser machining. Whilst all three images contain the same laser-machined structure, and hence also include the same random defects and debris, this translational augmentation process was repeated for many laser-machined structures, hence providing the neural network with information regarding different spatial positions, and different random defects and debris.

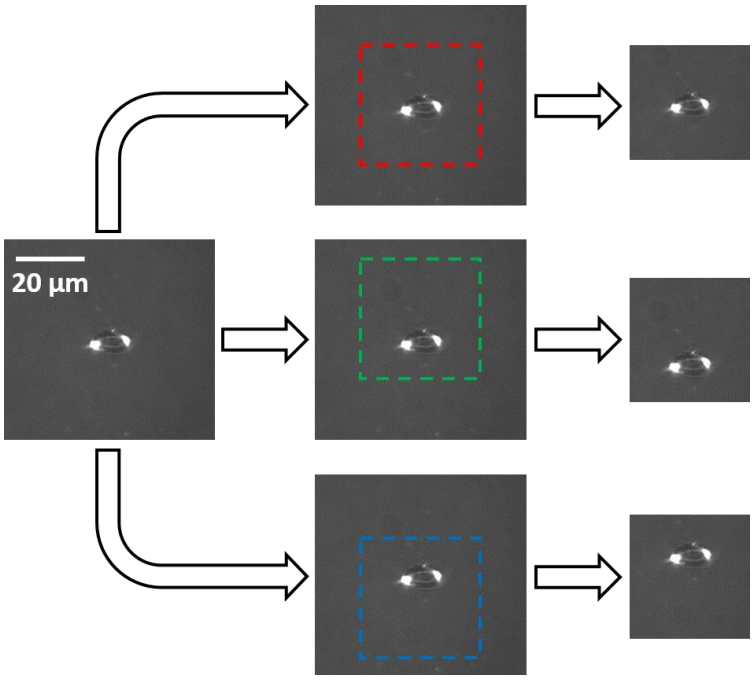

Figure 5-5 Data augmentation process, where the red, blue and green dashed squares are cropping windows, hence resulting in the appearance of a translation in the position of the laser-machined structure.

5.5 Identification of Beam Translation

In this section, the convolutional neural network was used to determine the translation of the centre of a laser-machined structure along one axis, directly from the information contained in the camera observation of the laser-machined structure. In Section 3.3, it is discussed that a convolutional neural network utilises filters as feature extractors to firstly transfer an input image to a set of feature maps. This is a different to, for example, a simple mapping function that map an

input image to an output array of fixed-size that can be interpreted in a meaningful way (e.g., Hash function). These feature maps are then mapped to an abstract representation of the input image, which can then be mapped to an interpretable output array (i.e., rotation angle or distance of translation). A feature extractor can be applied to any input images, including input image that are not observed during training. This scheme of mapping gives rise to an important capability of a convolutional neural network, namely that it possesses an ability to interpolate across inputs (see Section 3.2). In other words, a convolutional neural network does not need to be trained on all possible inputs to be able to map them, but instead only needs to be trained on a fraction of all possible inputs, where the learnt generalised mapping function can then be used to interpolate across the 'missing' inputs. In this section, this interpolation ability is demonstrated through augmenting a set of training materials to sets of training materials with different sizes.

There existed two sets of data that were collected in this experiment:

1) A small set of data that was augmented to a large set of data for training. This dataset will be referred to as the training dataset.
2) A set of data that was used to examine the performance of the convolutional neural network that was trained using the augmented training dataset. This dataset will be referred to as the testing dataset.

The training dataset contained 101 camera observations, each corresponding to one of the 101 laser-machined structures. The DMD mask was keep constant for the whole experiment, with a rotation angle $\theta = 0$ and translation distance of $d_x = 0, d_y = 0$. The training dataset was augmented along the axis of d_y, via data augmentation, in which the cropping window was shifted away from the centre of the image by integer numbers of camera pixels, ranging from 0 to 25 camera pixels with an increment of one camera pixel. Therefore, through applying the data augmentation, a small dataset, which consisted of 101 camera images of the elliptical laser-machined structures could be expanded to a larger dataset that consisted of a total number of 101 × 26 = 2626 camera images of the elliptical laser-machined structures. The convolutional neural network could then be trained on 101 different laser machined structures, each with 26 different translations. As per the discussion of the scheme of training in Section 4.4, 10 % of the images from the augmented dataset were randomly selected, and constituted a 'validation set', before the training commenced. The remaining 90 % of the augmented dataset was provided to the neural network in conjunction with their labels (i.e., the number of camera pixels shifted in the direction along the d_y axis) as the 'training set', and the convolutional neural network optimised its internal parameters in the direction of minimising the discrepancy between its predictions of the training set and the labels of the training set. It is important to realise the difference between the validation

dataset and the testing dataset. The validation dataset is used for monitoring of the training process (i.e., the neural network is trialled on the validation dataset during training), and hence can be used for optimisation of the training hyperparameters. The testing dataset is kept separate until the neural network is trained, before being applied to the network. Hence, the testing dataset is both unseen during training, and does not influence the training process. As per the discussion Section 4.4 and Section 4.5, the validation set was fed to the convolutional neural network without the company of their labels, which suggests that the convolutional neural network had to predict the translations in the direction along the d_y axis of the images from the validation set without knowing their ground truth (i.e., labels). Therefore, the discrepancy between the performances of the convolutional neural network on these two datasets determined the degree of overfitting, which was used as a merit to determine the generality of the convolutional neural network (i.e., its ability to determine the translations in d_y of unseen camera observations of elliptical laser-machined structures).

The testing set, on the other hand, contained 202 camera observations, each corresponding to one of the 202 elliptical laser-machined structures, where these elliptical laser-machined structures were translated by a physical modification to the DMD pattern displaying the elliptical light mask. In other words, the training dataset contained images of laser-machined structures that had the appearance of being translated through the cropping of the camera images. However, critically, the testing dataset contained images of laser-machined structures that were physically translated via translation of the elliptical pattern on the DMD itself. The training data therefore contained artificial translations through augmentation, but the testing dataset contained real-world physical translation through modification to the DMD. This motivation here was to train the neural network on augmented data (hence significantly reducing the amount of training data needed to be collected), whilst demonstrating that the trained neural network could still be applied to real-world translations. For the testing dataset, modifications were achieved through the translation of the elliptical light mask away from the centre of the DMD micro-mirror array, in the direction along the axis of d_y, by an integer number of micro-sized mirrors, ranging from 0 to 100 micro-sized mirrors with the increment of one micro-sized mirror (i.e., two camera observations per step of micro-sized mirror translated). In can be noted, that the training dataset contained 26 translation steps, whilst the testing dataset contained 101 translation steps. This is a consequence of the difference in scaling factors between the camera images (for the 26 translation steps) and the DMD array (for the 101 translation steps). Translating the elliptical light mask by four micro-sized mirrors on the DMD resulted in a translation of the elliptical laser-machined structure by approximately (but not exactly) one camera pixel. Therefore, the testing dataset covered positional modifications in the direction along the axis d_y corresponding to between 0 and 25 camera pixels on the

workpiece. This difference in scaling had an interesting consequence, namely that real-world translations of the elliptical pattern by a single DMD mirror corresponded to approximately ¼ of a camera pixel, and hence the testing dataset could be used to quantify the capability of the neural network to detect changes in position that were a fraction (i.e., 1/4) of a single camera pixel. As discussed above, it is important to reiterate that the testing dataset, which was composed of camera observations of the elliptical laser-machined structures that were laser machined through adjusting the position of the elliptical light mask in reference to the centre of the DMD micro-mirror array, was collected only for examining the performance of the convolutional neural network that was trained with an augmentation dataset. And thus, this set of data was not used as training materials, nor in the validation set. Including the images from the testing dataset in the validation dataset could lead to 'cherry-picking' of the parameters in the convolutional neural network (i.e., accidently during training the convolutional neural network could perform very well on the testing set, but this would not be an indication that the convolutional neural network could also perform well on other collected data). This separation of datasets was therefore an intentional decision, in order to allow the quantification of the capability of the trained neural network to learn a general solution (rather than the memorisation of training data).

In order to visually demonstrate the ability of the convolutional neural network to interpolate input data, six instances of the convolutional neural networks were trained, each of which was trained with a training dataset that was augmented to a different degree. The total number of possible degrees of augmentation equalled the total number of possible camera pixel shifts (i.e., 26) in the direction along the axis of d_y. The six instances of convolutional neural networks were trained with their training materials augmented to 1, 2, 3, 4, 5 and 26 of the total 26 points of augmentation respectively. For instance, in the training where only 1 point of augmentation is applied to the set of training materials, this set only contains 101 camera observations of the elliptical laser-machined structures that are visually shifted 13 camera pixels from the image centres along d_y, and therefore the total size of this set of training images is 101; whereas in the case for training data with 3 points of augmentation, the 101 camera observations of the elliptical laser-machined structures are shifted 0, 13 and 25 pixels from their image centres along d_y respectively, and therefore the size of this augmented set of training images is 101 × 3 = 303. The trained six convolutional neural networks were then used to determine the translations along d_y in the testing set, and the results are shown in Figure 5-6. The convolutional neural networks that utilised 1, 2, 3, 4, 5 and 26 points of augmentation to expand their respective training set are shown in a), b), c), d), e) and f) respectively. The translation positions that were augmented are shown as a large blue hollow circle, and the determined results (i.e., from the testing dataset) are shown as small red hollow circles. The overall prediction accuracies were characterised with the coefficient

of determination (R^2 defined in Section 4.4 equation 4-1) and the Root Mean Square Error (RMSE), which is defined by

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}(Prediction_t - GroundTruth_t)^2}{N}} \qquad 5-1$$

From Figure 5-6, it is observable that determined results (i.e., red circles) have a tendency of clustering around the points of augmentation (i.e., blue circles). This is likely due to the ability of the convolutional neural network to interpolate inputs, which enables the convolutional neural network to become a generalised model that can be applied to inputs that are not included in the training set. The results indicate that the convolutional neural network had a strong preference to interpolate unseen inputs closer to values seen during training, and hence clustering is observed around the blue circles. This also explains the reason why the accuracy of the convolutional neural network increased with the points of augmentation — more points of augmentation mean more accurate interpretations of the unseen inputs. The results from Figure 5-6 also suggests that, at least for the demonstrated laser machining experiment, the data augmentation approach can be applied to a small dataset to significantly reduce the reliance on the data collection. Interestingly, from b) to e) in Figure 5-6, it is shown that even for the sets of training materials that were not augmented to a great extent, the convolutional neural networks were still capable of achieving satisfactory levels of accuracies in identifying the translations of the elliptical laser-machined structures along d_y. This clearly demonstrates the effectiveness of the convolutional neural network in mapping sensory signals (e.g., camera observations) to interpretable values, in the specific case for translation of the observed feature.

It is important to note that the camera observations of the elliptical laser-machined structures contained random sets of features, and hence each structure was unique in appearance. These features comprised, for example, randomly positioned debris and small fragments (originated from the shock wave of the incident pulse) and unique surface morphology. Whilst deliberately re-using the same camera observation of the same elliptical laser-machined structure multiple times (i.e., data augmentation) significantly reduces the reliance on the data collection, it also resulted in the convolutional neural network training repeatedly on laser-machined structures with identical randomised features, effectively yielding a bias that maps these features to the output of the network. Therefore, in this case, and in the more general case of training data augmentation, there is a trade-off in terms of reducing reliance on the data collection and introducing biases. However, at least for the demonstrated experiment, it is evident from Figure 5-6 that the convolutional neural networks, which were trained with the augmented datasets, were indeed able to precisely and accurately determine the translations of the elliptical laser-machined

structures that were physically located away from the centre of camera observation (i.e., testing set). Interestingly, the approximated scale of a camera pixel was ~360 nm and the highest accuracy that the neural network was able to achieve was ~240 nm, indicating that the neural network was able to identify translations that were smaller than a single camera pixel. The cause of this sub-resolution accuracy likely originated from the property that light incident on a single camera pixel will result in 100 % of that intensity being recorded in a single pixel, whilst light that is incident across two camera pixels will result in 50 % of that intensity being recorded in each of the two pixels. Similar sub-resolution detection algorithms have frequently been discussed elsewhere in the literature.

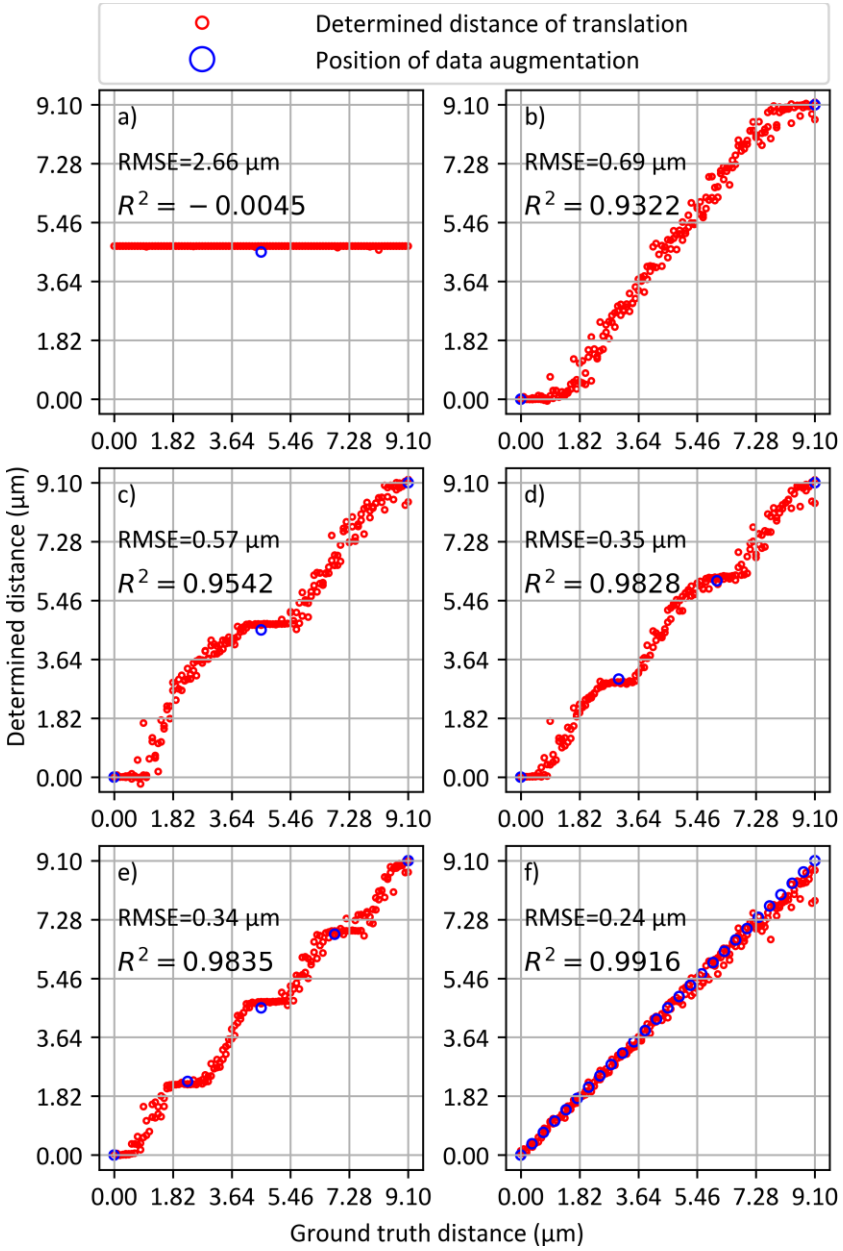

Figure 5-6 Accuracy of the convolutional neural networks, each trained with different degrees of augmentation.

5.6 Identification of Beam Translation and Rotation

In this section, a similar experiment to the experiment in Section 5.5 is demonstrated. However, in this case, the complexity of the experiment is increased — two convolutional neural networks were tasked to detect the translations of the elliptical laser-machined structure along both d_x and d_y axes respectively, while a third convolutional neural network was tasked to detect the rotation angle θ of the elliptical laser-machined structure. In other words, this section describes the extension from the identification of the translation in one axis, to the simultaneous identification of both the translation in two axes and the rotation. Similarly, in this experiment, the data augmentation was applied to a set of training material through cropping the camera observations of the elliptical laser-machined structures along both d_x and d_y axes, in order to approximate the corresponding physical translations of the elliptical laser-machined structures. However, since the experiment setup employed a side illumination, all camera observations of the laser-machined structures had features that varied with their respective rotation angles (e.g., shadows and highlights). It was therefore not viable to augment the rotation angle in this experiment. Therefore, the collected training dataset was formed of 180 camera observations of 180 elliptical laser-machined structures. Each of the elliptical laser-machined structure were rotated about the d_y axis of the DMD micro-mirror array by a unique integer angle, ranging from 0 to 179 degrees, with an increment of one degree. This training dataset was then augmented along the d_x and d_y axes, for 26 positions, giving a total number of 180 × 26 × 26 = 121,680 images. As discussed previously in Section 5.5, 90 % of these images constituted the training dataset, and the remainder 10 % of these images constituted the validation dataset. Also as before, a testing dataset was collected, where laser-machined structures that were both translated via shifting the DMD pattern, and rotated via rotating the DMD pattern, were collected. Therefore, the training dataset was augmented for position and physically modified for rotation, and the testing dataset was physically modified for both position and rotation. In total, there were 277 camera observations of the elliptical laser-machined structures collected for the testing dataset, where each of the elliptical laser-machined structures were machined with the elliptical light mask that was randomly modified, with a random translation along both axes and a random rotation (d_x, d_y, θ)). The random modifications to the elliptical light mask were determined by three uniform distributions separately and independently (i.e., $d_x = \mathcal{U}_{[0,100]}, d_y = \mathcal{U}_{[0,100]}, \theta = \mathcal{U}_{[0,179]}$). Three convolutional neural networks were used to determine the translation along d_x aixs, d_y axis and the rotation θ respectively.

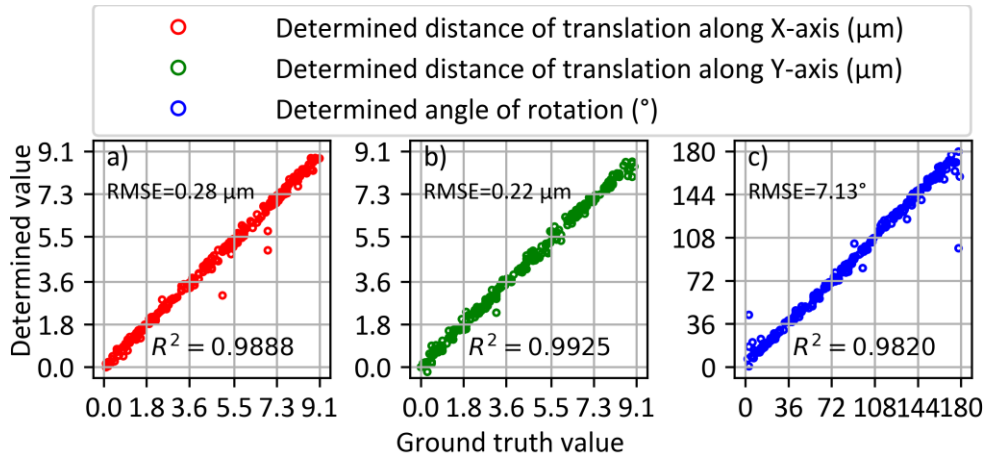

Figure 5-7 Accuracies of the convolutional neural networks for the testing dataset.

Figure 5-7 a), b) and c) show the accuracies of the trained convolutional neural networks in determining the translations along d_x axis (red hollow circles) and d_y axis (green hollow circles) and the rotation angles (blue hollow circles) of the testing dataset, respectively. This result suggests that the trained convolutional neural networks were able to determine the translations along d_x and d_y axes with accuracies of 280 nm and 220 nm respectively, both of which was below the scale of a single camera pixel (i.e., 360 nm). For the third convolutional neural network that was tasked to determine the rotation angle, an important modification to the cost function of the convolutional neural network was needed; instead of directly determining a rotation angle based on the camera observation of a rotated elliptical laser-machined structure, the convolutional neural network determined the sine and cosine of the rotation angle. This minor change was critical to the accuracy of the convolutional neural network in determining the rotation angle. Owing to the two axes of symmetry of the chosen elliptical shape, a large and incorrect measurement of the discrepancy between the directly determined rotation angle and the ground truth rotation angle can be given by the HMSE objective function (see Section 3.2.2) if the ground truth rotation angle is near 0 degree or 179 degrees. For instance, if the ground truth rotation angle is 179 degrees rotation and the directly determined rotation angle is -1 degree, the HMSE objective function (see equation 3-2) would give a discrepancy measure of $(1° - 179°)^2/2 = 15{,}842$, while the actual discrepancy, taking into the symmetry, is $(1° - (-1°))^2/2 = 2$. Through determining the sine and cosine of the angle, on the other hand, HMSE objective function gives $(\sin 1° - \sin 179°)^2/2 \approx 0.0$ and $(\cos 1° - \cos 179°)^2/2 \approx 1.9$, while the actual discrepancy, factoring in the symmetry, is $(\sin 1° - \sin -1°)^2/2 \approx 0.0$ and $(\cos 1° - \cos -1°)^2/2 \approx 0.0$. From Figure 5-7 c), it is observable that the accuracy for the convolutional neural network in determining the angle of rotation was approximately 7.13 degrees. Although the modified cost function avoided erroneous returns with large magnitudes that may lead to disastrous effects such as the exploding gradient, it can still be seen that the erroneous determinations at around 0 and 180 degrees contributed significantly to the total error. It is therefore anticipated that the accuracy in the determination of the rotation angle of a laser-

machined structure with a symmetrical shape could be further improved with a different cost function.

In practice, each camera observation was simultaneously sent to all three trained convolution neural networks, and therefore three outputs that corresponded to the two translations and one angle were determined separately, but at the same time. Whilst this experiment only demonstrated detections of the translation and rotation of laser-machined structures, this technique could also be extended to the detection of other parameters, for instance, pulse fluence and beam shape, with the overriding purpose of enabling real-time monitoring of laser machining. In addition, whilst identification of the rotation of elliptical beam may appear rather prescriptive, there are many cases where fluctuations between optical transverse modes can result in rotational changes to a beam spatial intensity profile. Admittedly, however, the primary motivation for identification of the angle in this experiment was to provide a more general demonstration that a neural network can detect morphological changes in the beam spatial intensity profile.

5.7 Monitoring and Control of Machining Depth

In this section, a feedback loop that can adaptively apply laser pulses to completely machine through a layer of thin film with a varying thickness, without damaging the underlying substrate, is demonstrated. Whilst applying an excessive number of incident pulses to a multilayer structure, where only the top layer should be machined, may indeed ensure the complete machining of the top layer, it may also of course damage the underlying substrate. In addition, this approach of applying an excessive number of laser pulses may result in inefficiencies regarding total machining time and indeed energy efficiency. Ideally, a real-time system capable of adjusting the number of applied incident pulses in accordance with the thickness of the thin film would be applied. An example of such could be a thickness-measurement technique (e.g., surface profilometer or phase-shift interferometry) could be employed in order for a controller to determine the thickness of the thin film in real time. A different approach, which does not require any additional apparatus, and which is the approach demonstrated here, is through observing the laser-machined structures on the thin film in real-time via a camera, and adjusting the number of incident pulses adaptively via a convolutional neural network. In this section, the convolutional neural network was employed to

1) determine the number of pulses there were needed for penetrating the thin film, based on the camera observation of the laser-machined structure.
2) halt the following incident laser pulses in real-time when the thin film is already machined through.

In the experiment, a convolutional-neural-network-facilitated controller was incorporated to a feedback loop in the devised system shown in Figure 5-1. This controller was employed to determine whether further laser pulse was required to be incident at the workpiece, based on the camera observations of the thin-film-deposited workpiece. A camera observation of the thin-film-deposited workpiece was made following each incident pulse. This camera observation was then sent to the controller, which would determine the number of remaining pulses until the thin copper film was completed machined through. If this determined number was a positive number, the controller would apply a subsequent pulse to the thin-film-deposited workpiece; if the determined number was zero or a negative number, the controller would halt further incident pulses. Here, if the neural network determined a value with a negative sign (i.e., -X), this corresponded to a prediction that the sample had been over-machined by X pulses. In order to ensure that the thickness of the deposited copper varied across the sample, the thin copper film was coarsely polished with a piece of sandpaper (which induced approximately ± 100 nm variation in the thickness of the cooper film). This was done due to the concern that the controller could learn to determine the number of remaining pulses blindly (without any reference to the observations of the workpiece), and still achieved a high accuracy in determine the number of remaining pulses, if the thickness of the copper film was nearly a constant. Owing to the introduced variations in the thickness of the copper film, the controller needed to assess the observation of the laser-machined structure on the workpiece to determine the number of remaining pulses to penetrate through the thin copper film.

To generate training data, 156 elliptical hole structures were laser machined on the workpiece, with each of the hole structure being machined with 50 consecutive incident pulses. During this experiment the DMD pattern was not changed. These $50 \times 156 = 7800$ camera observations of the elliptical laser-machined structures constituted the training data for the convolutional neural network to learn how to determine the remaining number of pulses to penetrate the thin copper film. And, as per the training protocol discussed in Section 5.5, this dataset was split into the training dataset and the validation dataset, with the size ratio of 9 to 1, respectively. As fewer than 50 consecutive pulses were required to machine through the top layer of the sample in each case, there existed 156 recorded observations of laser-machined structures before, during and after the point at which the thin copper films were completely machined. These recorded observations were manual inspected and labelled, where the point of breakthrough was labelled ‘0’. The corresponding structures of these observations were verified via reflectivity and transmission measurement using an optical microscope, and through surface height measurement via the interferometric profilometer. With this information acquired, each of the 7800 recorded camera observations of the machined structures could be labelled in accordance with their

respective differences in number of applied pulses in reference to their respective moments of penetration (i.e., -3, -2, -1, 0, 1, 2, 3 pulses). Once trained, the convolutional neural network was integrated into the laser machining experiment setup as the controller, and then tested in real-time on fresh (i.e., not machined) regions of the sample.

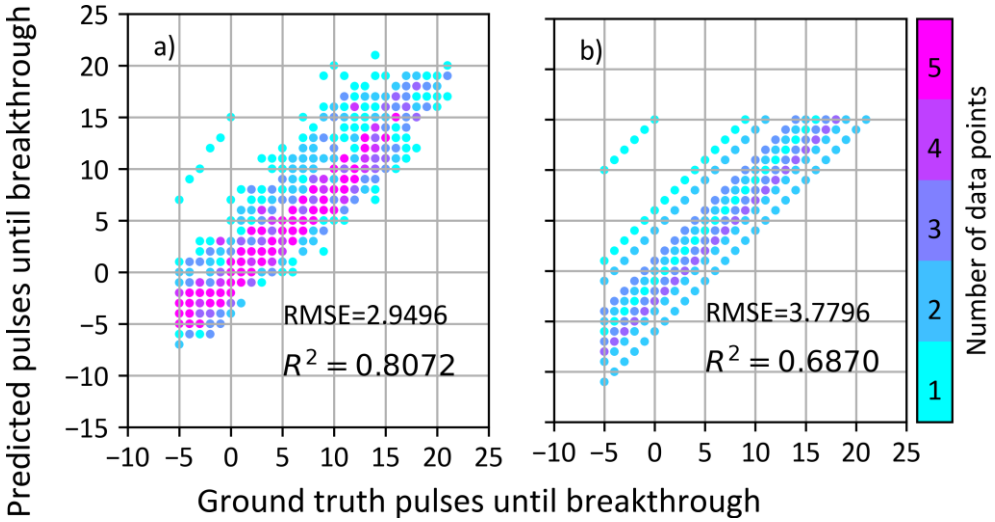

Figure 5-8 Experiment results of a) the neural network controller and b) naive guesses using the average number of 15 pulses.

The controller that incorporated the trained convolutional neural network was then tested in a laser machined experiment, and the results are shown in Figure 5-8 a). A total number of 21 structures were machined with the convolutional neural network determining the number of remaining pulses in real time. It took, on average, 15.33 pulses to penetrate the thin copper film, and the accuracy in determining the remaining number of pulses until breakthrough achieved by the convolutional neural network was 2.9496 (pulses). For a quantitative comparison, Figure 5-8 b) equivalent result if the real-time controller was not used and instead an average material thickness assumed that 15 pulses required to machine the copper film. This naïve guess gives an accuracy of 3.7769 pulses. Therefore, the convolutional neural network is shown to have improved the accuracy, compared with that of adapting a naïve guess strategy, by ~28.05 %. Of particular interest is that the Figure 5-8 a) shows that the neural network is able to predict, to a high accuracy, the number of pulses remaining, even when ~15 pulses are still required. This capability unlocks the potential for modification of the laser machining parameters in the final stages of machining.

5.8 Conclusions

In conclusion, two applications of convolutional neural networks for control of femtosecond laser machining were presented. Firstly, this network architecture was shown capable of identification of two beam modifications, namely translation of the beam and rotation of the beam, via

observation of the laser machined structures. In practice, unexpected translation of the beam position could occur through random misalignment in the laser and/or from optical misalignment between the laser and the workpiece. Unexpected rotation of the beam may occur due to mode changes and/or optical misalignments. It was shown a convolutional neural network could detect translations in the beam position to a precision that was smaller than the size of a single camera pixel. Secondly, a convolutional neural network was integrated into a closed-loop feedback system to cease laser machining at the point where the top layer of a multilayer structure was machined, despite the thickness of the top layer not being known. These results indicate that convolutional neural networks offer a tantalising potential for real-time monitoring and control of high-precision laser machining.

Chapter 6 Controlling Optical Tweezers using Reinforcement Learning

Reinforcement learning offers the capability for solving complex challenges through trial and error. In this chapter, reinforcement learning is used to enable automated control of a set of translation stages, to move a microparticle to a target location, via the optical tweezers effect, whilst avoiding other particles and obstacles. The well-known optical tweezers effect (as described in section 2.4) can be used to 'trap' a particle in a laser focus, the microparticle can then follow the movement of that laser focus. This chapter describes the progression from training of the reinforcement learning agent in a virtual environment, the application in the real-world (physical) environment, and finally the application in a 'hybrid' environment that is formed by both virtual and physical components.

For the work in this chapter, the author created, optimised and trained the reinforcement learning agents, collected the experimental data, and performed all analysis. The author proposed the discount factor approach to address the partially observability that was induced by the limited field of view. The experimental setup was jointly constructed by the author, Dr Matthew Praeger and Dr James Grant-Jacob. The virtual environment was constructed by the author with the help of Dr Matthew Praeger. The automation of the translation stages was completed by Dr Matthew Praeger.

6.1 Introduction

In this chapter, a reinforcement learning agent was incorporated into an optical tweezers experimental setup. This setup was able to steer a microparticle to a predefined target location through non-contact optical force, via translation stages that were controlled by a reinforcement learning agent. Figure 6-1 schematically depicts the concept of the experiment, where a free-moving microparticle was trapped via the optical tweezers effect, and was then transported from its initial position to a predefined target position without any collisions with other microparticles. The moving path of the laser-trapped particle was partially determined from the top-down camera observation of the sample, which was centred at the laser-trapped microparticle (additional factors in path determination include stage position data and a vector pointing towards the target location). This top-down camera observation had a limited field of view, which imposed a restriction on visibility of the predefined target position. However, since the limited field of view of the camera moved with the laser-trapped microparticle (as the camera observation was centred at the laser-trapped microparticle), the predefined target position became visible to the decision-making

reinforcement learning agent when the laser-trapped microparticle was near the predefined target position. This proposed system could be used to transport a fragile micro-sized part from one place to a different place on a workpiece, where the part could be used to assemble a larger micro-sized device.

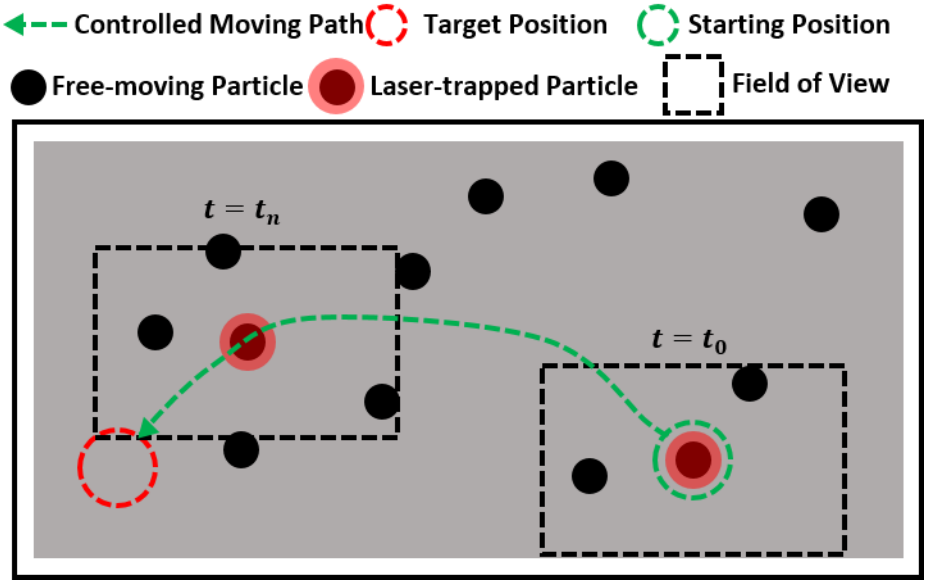

Figure 6-1 Schematic of the optical tweezers experiment.

As per the discussion in the previous chapter (Section 3.4), a virtual environment was used in this experiment. This virtual environment approximated the physical environment in order to allow the reinforcement learning agent to learn pathfinding and collision avoidance capabilities, without the need for data collection in the physical environment. In addition to the physical and virtual environments, a hybrid environment was also demonstrated, where barriers that were made of fictional microparticles were added to the observations of the real-world microparticles. This hybrid environment enabled the laser-trapped microparticle to perform collision avoidance on the virtual obstacles as well as the real-word microparticles, while heading towards a predefined target position. Whilst there exit similar studies that aim to control an object to simultaneously perform pathfinding and collision avoidance, these studies are usually conducted in large-scale environments (Alwala and Mukadam, 2020, Mukadam et al., 2018).

6.2 Experimental Setup

A Thorlabs optical tweezers education kit (*Thorlabs* EDU-OT2/M) was used as the physical environment in this experiment. A picture of the experimental setup and schematic of the beam path are shown in Figure 6-2. A 658 nm, 40 mW laser diode (*Thorlabs* L658P040) was used as the laser source, which was collimated (*Thorlabs* LTN330-A), expanded (*Thorlabs* LA1074-A and LA1509-A) and focused (*Zeiss* Microscope Objective 63×, N.A. 0.8) onto the sample. The sample was

mounted on a sample holder and could be translated independently along two perpendicular axes (i.e., X- and Y-axis) in the plane that was orthogonal to the direction of the incident laser energy. The XY motorised stages (with *Thorlabs* KDC101 controllers and Z812B DC servo actuators), were responsible for the sample holder translations along these axes and had a maximum travel length of 12 mm, a maximum velocity of 2.6 mm/s and a minimum step size of 0.05 μm. In this experiment, the velocities of these two motorised stages were controlled directly by the reinforcement learning agent. The sample contained silicon dioxide microspheres (5 μm ± 100 nm in diameter, *Sigma Aldrich* 44054-5ML-F) diluted in distilled water. The sample solution (concentration ~0.125 % solids by mass) was sandwiched between a glass microscope slide and a cover slip, with the surrounding areas sealed by adhesive gasket tape. The sample holder was manually adjusted along the Z axis (i.e., the axis of the incident laser) at the start of the experiment, to ensure that the beam focus enabled the microsphere trapping force in the plane of the sample. A CMOS camera (*Thorlabs* DCC1645C, resolution of 1024 × 1280) and lens (*Thorlabs* LB1676) were positioned above the sample, producing a microscope system that allowed for simultaneous monitoring of the laser-trapped microparticle with the help of a beam splitter (*Thorlabs* BS022).

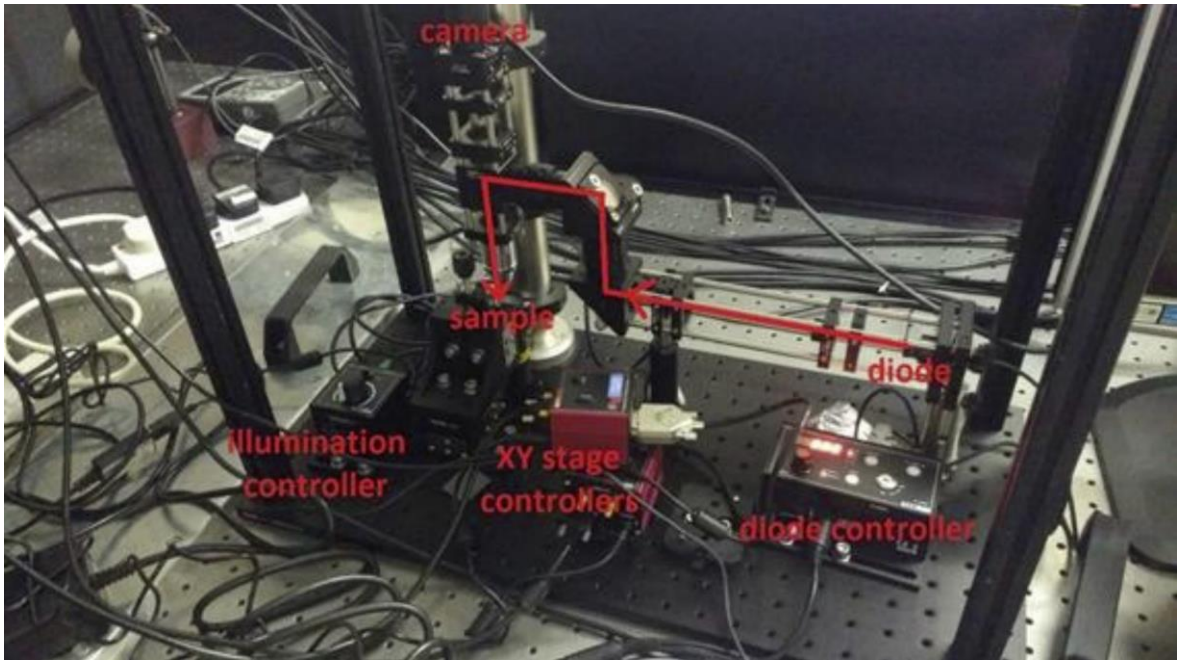

Figure 6-2 Schematic of the experimental setup and the beam path.

6.3 Neural Network Architecture

Table 6-1 and Table 6-2 present the connectivity of the neural networks that were used as the actor and critic respectively, in the TD3 algorithm (See Section 3.5.1). The TD3 algorithm employs one actor and two critics (the actor is updated based on the lower value estimate made by the two critics, i.e., clipped, to minimise overestimation errors). TD3 further reduces error growth by

making use of target networks to allow delayed update of parameters, through Polyak averaging of the parameters inside each network ($\theta_t = \gamma \cdot \theta_t + (1-\gamma) \cdot \theta_{t-1}$, for update rate $0 < \gamma < 1$). The connectivity of these neural networks was structurally similar to the connectivity of the neural network employed in (Mnih et al., 2015), where the design philosophy could be summarised by 'doubling the width when halving the length' (i.e., double the number of filters each time the layer size is reduced by a convolution with a stride of 2). Since the actor controlled the velocities of two orthogonal motorised stages, the output of the actor was of the shape 1 × 2. The critics, on the other hand, mapped state-action pair to a state-action value, and therefore the output of the critics was of the shape 1 × 1. Importantly, in this study, owing to the need for the reinforcement learning agent to observe two types of inputs (which will be discussed further in Section 6.4) namely, sensory input and positional input, the neural network concatenated the abstract representation of the sensory input and the raw positional input after the second fully connected layer. This enabled the reinforcement learning agent to determine an action jointly based on both the sensory input (i.e., the camera observation of a laser-trapped microparticle) and the positional input (i.e., the vectorised displacement between a laser-trapped microparticle and a predefined target position). In other words, the observation of a state s_t was composed of two components, namely the sensory input and the positional input. Owing to the difference in the definitions of the actor function $\mu(s)$ and the critics function $Q(s,a)$, the action a was concatenated to the abstract representation of the sensory input in the critic but not in the actor. This led to a slightly different connectivity between the neural networks of the actor and the critics. Other parameters can be found in Table A 3.

Layer name	Input size (pixels)	Parameter (pixels)	Output size (pixels)
Input layer	1280 × 1024 × 3	-	256 × 320 × 1
Conv_1	256 × 320 × 1	8 × 8 × 32, stride of 4	64 × 80 × 32
Conv_2	64 × 80 × 32	4 × 4 × 64, stride of 2	32 × 40 × 64
Conv_3	32 × 40 × 64	3 × 3 × 64	32 × 40 × 64
Flatten	32 × 40 × 64	-	1 × 81920
Fully_Connected_1	1 × 81920	81920 × 512	1 × 512
Fully_Connected_2	1 × 512	512 × 64	1 × 64

Positional_Input	1 × 64	-	1 × 66
Fully_Connected_3	1 × 66	66 × 64	1 × 64
Actor	1 × 64	64 × 2	1 × 2

Table 6-1 Neural network architecture for the actor of TD3.

Layer name	Input size (pixels)	Filter size (pixels)	Output size (pixels)
Input layer	1280 × 1024	-	256 × 320
Conv_1	256 × 320	8 × 8 × 32, stride of 4	64 × 80 × 32
Conv_2	64 × 80 × 32	4 × 4 × 64, stride of 2	32 × 40 × 64
Conv_3	32 × 40 × 64	3 × 3 × 64	32 × 40 × 64
Flatten	32 × 40 × 64	-	1 × 81920
Fully_Connected_1	1 × 81920	81920 × 512	1 × 512
Fully_Connected_2	1 × 512	512 × 64	1 × 64
Positional_Input and Action_input	1 × 64	-	1 × 68
Fully_Connected_3	1 × 68	68 × 64	1 × 64
Critic	1 × 64	64 × 2	1 × 1

Table 6-2 Neural network architecture for the critics of TD3.

6.4 The Physical and Virtual Environments

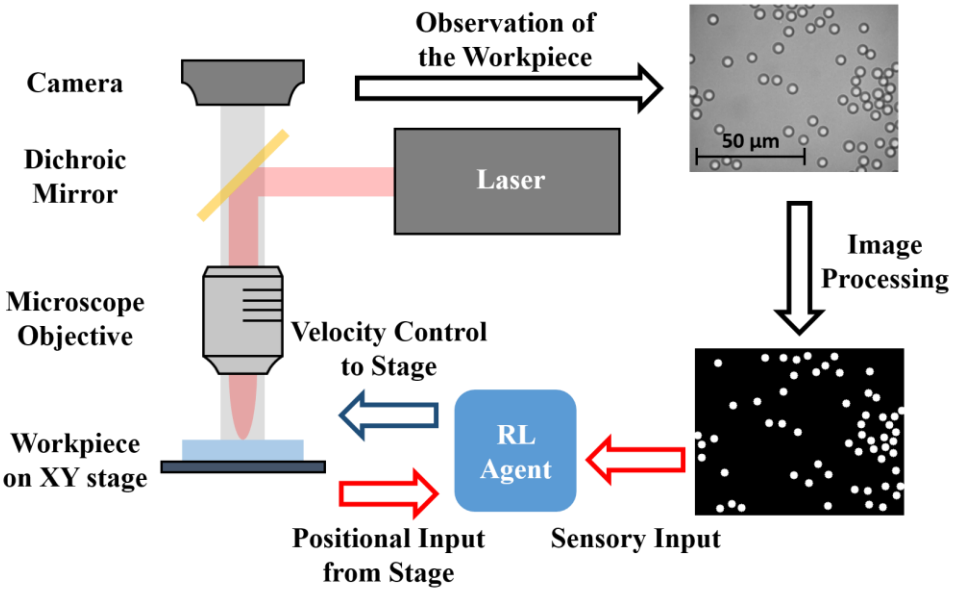

Figure 6-3 Flow diagram of the experiment rollout in the physical environment.

Figure 6-3 schematically depicts the rollout of the experiment in the physical environment in a flow diagram. The input to the decision-making reinforcement learning agent was composed of two pieces of information, which contained sufficient information concerning the virtual or physical environment for the reinforcement learning agent to determine an appropriate action, and these two pieces of information were:

1) The processed top-down camera observation of the laser-trapped microparticle, and its surrounding area (noted as the 'Sensory Input' in Figure 6-3).
2) A vector that contains two numerical values that represented the vectorised displacements along X and Y axes between the laser-trapped microparticle and the predefined target position (noted as the 'Positional Input from Stage' in Figure 6-3).

The action that was determined by the reinforcement learning agent (i.e., the chosen decision) was a vector that contained two numerical values, which were used to update the velocities of the X and Y motorised stages accordingly. This action was therefore used to control the translation of the laser-trapped microparticle. The control of the motorised stages occurred at discrete time steps. Each step was composed of the following sub-steps:

1) At the start of each time step, a camera observation of the laser-trapped microparticle and its surrounding area was made. This observation was processed with an image processing procedure, which segmented every visible microparticle on this observation (including both the free-moving microparticle and the laser-trapped microparticle) from the background.

The segmentation process was based on the Hough Transform, as described later. This processed camera observation is referred to as the sensory input.

2) The displacement between the predefined target position and the laser-trapped microparticle (i.e., the centre of the camera observation) is calculated as the positional input.
3) Both the sensory and the positional inputs, which are obtained in the current step, are simultaneously inputted to the reinforcement learning agent.
4) The reinforcement learning determines an action that is composed of two numerical values. These two values are then inputted to the two motorised stages, the X and Y axes respectively.
5) Each motorised stage adjusted its velocity based on the received numerical value, and this velocity was maintained for the period of time until the next time step started.

These steps were repeated until one of the following termination conditions was met:

1) The laser-trapped microparticle collided with a free-moving microparticle (i.e., a second microparticle was trapped by the laser).
2) The laser-trapped microparticle reached the predefined target position.
3) A total number of 1000 steps had passed and none of the other termination conditions were met (e.g., the laser-trapped microparticle moves back and forth).

6.4.1 Image Processing of Sensory Input

The image processing procedure provides a reliable means to segment the areas where microparticles reside from the background areas where microparticles do not exist in the camera observation. This image processing procedure consists of the following image processing steps:

1) Image smoothing of the camera observation via Gaussian blurring.
2) Image adaptive thresholding to the smoothed image in order to segment it and extract the features of microparticles.
3) Image erosion in conjunction with image dilation to remove small noise features and separate groups of microparticles.
4) Hough transform to determine the coordinates of potential microparticle features.
5) Filtering of features identified by the Hough transform, excluding those with circularities or sizes are below certain thresholds.

A flow diagram of the image processing procedural is shown in Figure 6-4. Although the sample is sandwiched between a glass slide and a covering slip, it is possible for the visuals of microparticles

to vary significantly during the translation of a laser-trapped microparticle due to small variations in the position of the particles relative to the focal plane. A reliable segmentation algorithm that is capable of segmenting microparticles at different offsets from the focal plane is therefore critical to the success of the experiment, in the sense that it would be possible for the reinforcement learning agent to collide the laser-trapped microparticle with a free-moving microparticle that is not visible to the agent. The adopted image manipulation methods in this image processing procedure are all of common use and are available on many libraries across many operating systems. In this work, the image processing procedure was built in *Python*, using the *OpenCV* library (Bradski and Kaehler, 2000). The time taken for transferring a single camera observation (with a native resolution of 1024 × 1280) to a processed sensory input (with a resolution of 256 × 320) was approximately 0.006 seconds in a low-end desktop configured with a 4th generation Intel i5 CPU.

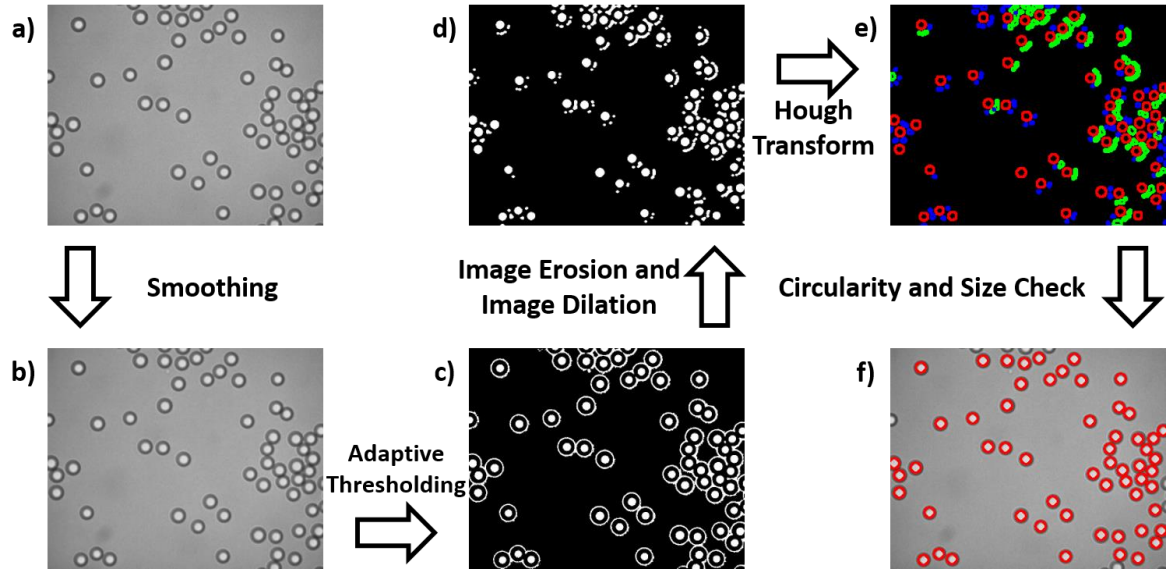

Figure 6-4 Flow diagram of the image processing procedure. In e), the detected (enclosed) shapes with circularities and sizes below a predefined threshold are ruled out. In the figure, shapes that are excluded due to the circularity check are marked in green, and shapes that are excluded due to the size check are marked in blue. Note that red hollow circles are overlaid with the camera observation in the final output (i.e., f)); this is simply to demonstrate the accuracy of the procedure. (For the actual processed image sent to the neural network, solid circles are rendered in a blank image, as shown in Figure 6-3).

6.4.2 Calculation of Positional Input

In addition to the sensory input, which is processed from the camera observation, a positional input is also inputted to the reinforcement learning agent simultaneously. This positional input is a

vector with a direction corresponding to the target position from the current position, calculated via:

$$\vec{k}= \frac{\vec{k}_{target} - \vec{k}_{current}}{\left|\vec{k}_{target} - \vec{k}_{initial}\right|} \qquad (6-1)$$

For instance, at the start of an experiment in the physical environment suppose that a microparticle is laser trapped at $\vec{k}_{initial} = (0\ mm,\ 3\ mm)$, with respect to the home position of the X and Y translation stages, and the target position is set at $\vec{k}_{target} = (4\ mm,\ 6\ mm)$. At a later time, if the microparticle is trapped at $\vec{k}_{current} = (2.5\ mm,\ 4\ mm)$ then the resulting positional input can be calculated according to equation 6-1 as

$$\vec{k}= \frac{(4 - 2.5\ mm,\ 6 - 4\ mm)}{\sqrt{(4 - 0\ mm,\ 6 - 3mm)^2}} = \frac{(1.5,\ 2)}{5} = (0.3, 0.4)$$

The positional input is, therefore, a vectorised displacement pointing at the target position from the current position of the laser-trapped microparticle, normalised by the initial distance between the target position and the position of the laser-trapped microparticle. The reinforcement learning agent utilises this positional input to infer the direction of the target position, with respect to the current position of the laser-trapped microparticle, enabling the reinforcement learning agent to move a laser-trapped microparticle towards the direction of the predefined target position, without actually needing to observe the target position via the camera. It is important to realise that the positional input is calculated by taking the difference between the current stage position and the target stage position through knowledge of the stage actuator absolute positions. The reinforcement learning agent does not have access to the absolute positions, and instead is provided with the vector describing the direction to the target. The absolute positions are defined in terms of displacement from the home positions of the X and Y translation stages.

6.4.3 Summary of Neural Network Inputs

To summarise, the reinforcement learning agent uses two types of input to determine the velocities of the motorised stages that translate the laser-trapped microparticle, namely the sensory input and the positional input. The sensory input, which is obtained from the camera, provides information with regards to the free-moving particles that are in close proximity to the laser-trapped microparticle, within the field of view of the camera; enabling the reinforcement learning agent to avoid collision between the laser-trapped microparticle and nearby free-moving microparticles. On the other hand, the positional input is calculated from the target position and the current position of the laser-trapped microparticle (scaled relative to the initial displacement

between the particle and target). This input provides information for the reinforcement learning agent to infer the direction of the target position with respect to the current position of the laser-trapped microparticle; allowing the reinforcement learning agent to move the laser-trapped microparticles towards the target position. It is therefore the responsibility of the reinforcement learning agent to extract necessary information from these two inputs, and to determine appropriate velocities in which the laser-trapped microparticle should be translated along the X and Y axes.

6.4.4 The Virtual Environment

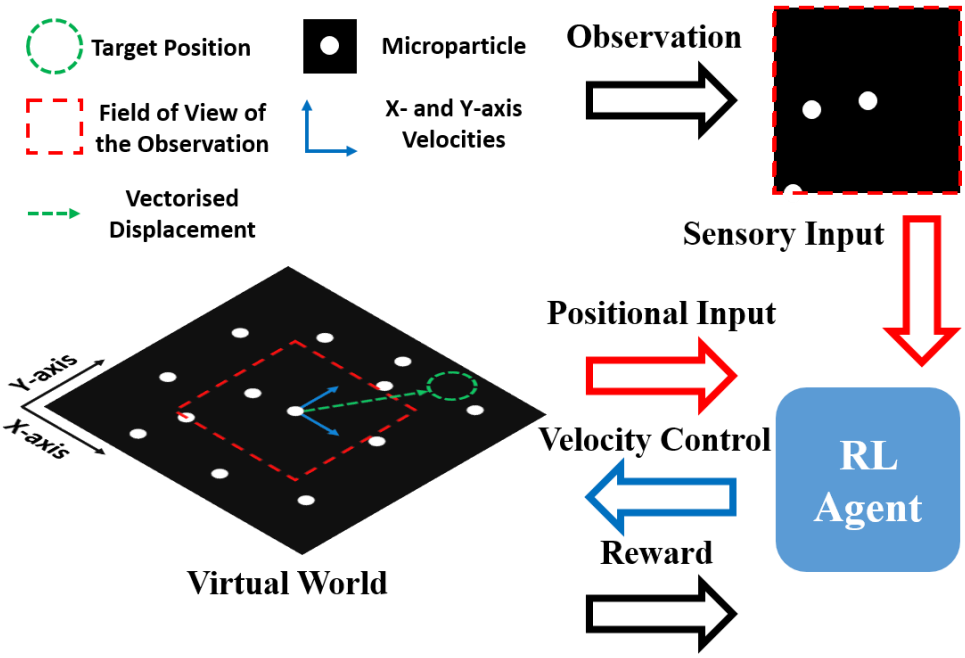

Figure 6-5 Schematic of the virtual environment.

Figure 6-1 schematically depicts the rollout of the experiment in the virtual environment. In order for the reinforcement learning agent to gain virtual experience and to become knowledgeable about the physical environment, the virtual environment is required to sufficiently approximate the physical environment. In the virtual environment, at the start of an episode, a random number of fictional free-moving microparticles (normal distribution, $\mathcal{U}_{[500,1000]}$) are scattered (i.e., assigned random positions) around a fixed size virtual world (0.278 mm by 0.278 mm), and a fictional laser-trapped microparticle that is controllable is positioned at the centre of this virtual world. The free microparticles are each randomly assigned a position (normal distribution, $\mathcal{U}_{[0,27.8]}$ in millimetres, separately initialised along X and Y directions), an initial velocity (normal distribution, $\mathcal{U}_{[-1.5,1.5]}$ in microns per second, separately initialised along X and Y directions) and an initial acceleration (normal distribution, $\mathcal{U}_{[-4,4]}$ in microns per second squared, separately initialised along X and Y directions). At each discrete time step, random accelerations are given to all free-moving

microparticles in order to simulate Brownian motion. These randomly applied accelerations are subject to a maximum allowed value (4 mm/s^2). In addition, the maximum velocities of all microparticles (including the laser-trapped microparticle) are capped at 0.015 mm/s. In order to approximate the sensory input, a top-down observation of the laser-trapped microparticle and its surrounding area (approximately 0.095 mm by 0.119 mm) is rendered at each step, taking into account the current position of all free particles currently within the field of view of the virtual camera. Similarly, the positional input is calculated by taking into account the position of a predefined target and the current position of the fictional laser-trapped microparticle at each time step. It is of great importance to emphasise that, as in the physical environment, these two inputs (sensory and positional) are the only source of information to that the reinforcement learning agent can access. Furthermore, the numerical values of the positions, velocities and accelerations of the fictional free-moving microparticles are not made available to the reinforcement learning agent. To reduce computational requirements, the virtual camera observations are rendered at a lower resolution of 256 × 320, instead of the native camera resolution 1024 × 1280.

The virtual environment was operated similarly to the physical setup in discrete time steps:

1) If the current step is the first step in an episode, a target position is randomly set at a position 1.2 mm away from the initial position of a laser-trapped microparticle (i.e., centre of the virtual world), and fictional microparticles are generated as per the description above.
2) The top-down observation of the fictional laser-trapped microparticle and its surrounding area is rendered, in order to simulate the camera image. Similar to the physical environment, this observation has a limited field of view.
3) The displacement between the position of the fictional laser-trapped microparticle and the target position is calculated.
4) The reinforcement learning agent determines an action based on its two types of input information. This determined action (i.e., the X and Y stage velocity values for the next time step) is then applied to the fictional laser-trapped microparticle.
5) The positions of all free-moving microparticles and the laser-trapped microparticle are evaluated and updated at once, assuming that a fixed period of time is passed (0.1 seconds).

The termination conditions of the physical environment apply to the virtual environment, namely a collision between the fictional laser-trapped microparticle and a fictional free-moving microparticle, the laser-trapped microparticle reaching the predefine target position, or the total number of steps exceeding 1000.

Collision detection between elastic objects in a simulation is, in itself, a topic of research; it can, for instance, be problematic determining collisions between two objects that are moving at high

velocities with data that has a relatively low sampling rate. Additional calculations (e.g., Bézier Interpolation) are generally needed for an accurate physical simulation of the collisions between multiple objects. However, owing to the fact that physical properties (e.g., velocity) of the microparticles after a collision are not of interest in the virtual environment (only collisions with the trapped particle are of interest) and the fact that the maximum distance by which the microparticles can move between two frames (i.e., 0.015 mm/s × 0.1 s = 0.0015 mm) is lower than the radius of the microparticles (i.e., 2.5 µm), these additional calculations are not adopted. Instead, the collision detection is achieved only though evaluating the distances between the laser-trapped microparticle and other free-moving microparticles.

6.5 Training in the Virtual Environment

Undoubtedly, the discrepancy in terms of the sensory inputs of the environment between the physical and the virtual environment should be minimised. In order to seamlessly apply a reinforcement learning agent, which is trained in the virtual environment, to a physical environment, the sensory inputs of these two environments should be indistinguishable to the reinforcement learning agent. However, due to the high resolution of the experimental CMOS camera (*Thorlabs* DCC1645C, resolution of 1024 × 1280), the function approximators of the reinforcement learning would require an unreasonably large memory (i.e., Random-Access Memory (RAM)) in order to be trained in this native resolution. Owing to this hardware limitation, the sensory inputs for both the physical and virtual environment was resized to a lower resolution, in order for the reinforcement learning agent to be trained on available hardware (i.e., graphics cards with RAM ≤ 16 Gigabytes).

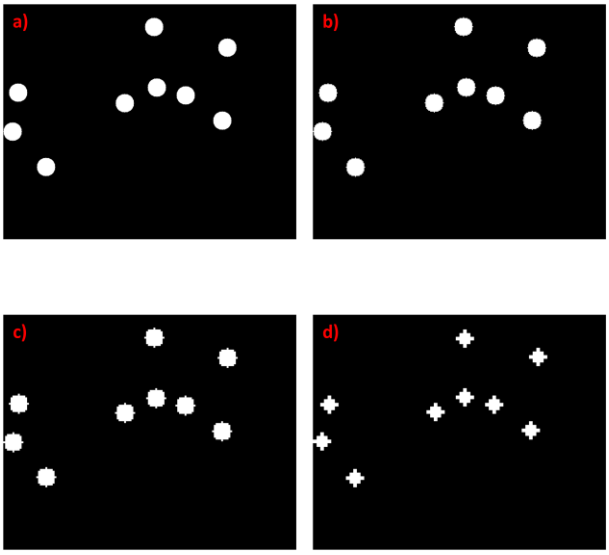

Figure 6-6 An observation from the virtual environment resized to a) 512 × 640, b) 256 × 320, c) 128 × 160 and d) 64 × 80.

Figure 6-6 shows a time step in an instance of the virtual environment rendered at four different resolutions, namely a) 512 × 640, b) 256 × 320, c) 128 × 160 and d) 64 × 80. Higher rendering resolution not only reduces the pixelisation of the microparticles, but also increases the fidelity with which the positions of the microparticles are represented. Since the sensory input is binary (i.e., only two colours are allowed and anti-aliasing is therefore not possible), and since the positions of microparticles (i.e., coordinates of centres of microparticles) are rounded to integer values (i.e., subpixel resolution is not used) therefore rendering at a higher resolution (i.e., more pixels for the same field of view) allows a more accurate representation of the positions of the microparticles. However, these benefits, which originate from increasing the rendering resolution, come at the expense of also increasing the computation requirements, as per the previous discussion. Figure 6-7 shows the training curves (moving average of undiscounted episodic reward, i.e., sum of rewards received throughout an episode (see equation 3-5), versus number of episodes, with a moving window of 100 episodes) under different rendering resolutions, namely 32 × 40 (blue curve), 64 × 80 (yellow curve), 128 × 160 (green curve), 256 × 320 (red curve) and 512 × 640 (purple curve). It is easily observable that the apex performance increase with the rendering resolution, and therefore the highest resolution (512 × 640) is theoretically the best choice for training of a reinforcement learning agent in the virtual environment (out of the resolutions plotted). Nevertheless, training a reinforcement learning agent at the resolution of 512 × 640 pixels requires a very large amount of RAM. During this study, hardware was limited to a *NVIDIA TESLA V100* GPU (i.e., 16 Gigabytes), and therefore a resolution of 256 × 320 pixels (which performed almost as well as 512 × 640) was chosen for the reminder of the training presented in this section.

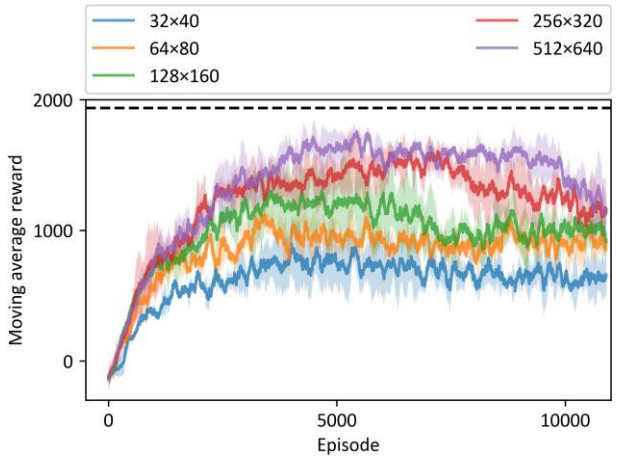

Figure 6-7 Training curves (i.e., undiscounted episodic reward versus number of training episodes) of the reinforcement learning agent in the virtual environment where the sensory input (i.e., virtual observation of the workpiece) is resized to 32 × 40 (blue line), 64 × 80 (yellow line), 128 × 160 (green line), 256 × 320 (red line) and 512 × 640 (purple line).

6.5.1 Reward Components

Constructing an appropriate reward structure is critical to incentivise a reinforcement learning agent to behave in a desirable way. A poorly designed reward structure very likely possesses exploitable loopholes that may lead to unexpected and undesirable behaviours. For this work, in the virtual environment, there exist four types of reward that can contribute to the total reward per step, namely

1) Goal reward of +1000, this is only given to a reinforcement learning agent when the reinforcement learning agent successfully translates a laser-trapped microparticle to the predefined target position, without any collision with other free-moving microparticle. This condition also terminates the episode.
2) Time step penalty of -1, this is a stepwise penalty that is imposed on the reinforcement learning agent, regardless of its action, at every step.
3) Collison reward of -100; this is only given to a reinforcement learning agent when it manoeuvres the laser-trapped microparticle to have a collision with one of the free-moving microparticles. This condition also terminates the episode.
4) Carry reward, this is a reward granted to a reinforcement learning agent at every step, with a magnitude equal to the number of pixels by which the reinforcement learning agent moves the laser-trapped microparticle towards or away from the predefined target position (i.e., The direction determines the sign of this reward, moving towards the predefined target grants the reinforcement learning agent a positive reward, and vice versa).

The primary motivation behind this reward structure is to encourage pathfinding and obstacle avoidance. Pathfinding can be encouraged through granting a high magnitude positive reward when the reinforcement learning agent successfully translates a laser-trapped microparticle to the predefined target, and obstacle avoidance can be encouraged through a high magnitude negative reward when the reinforcement learning agent collides with another free-moving microparticle. In addition, the carrying reward incentivises the reinforcement learning agent to move towards the predefined target pattern, while the time step penalty incentivises a reinforcement learning to reach the predefined target position using as few steps as possible. The impact of each reward type to the learning process of a reinforcement learning agent is shown in Figure 6-8, where the training curves of five reward combinations are plotted. In four of the five training curves, one of the four reward types was not granted to the reinforcement learning agent, namely without the carry reward (blue line), without the collision reward (yellow line), without the time step reward (green line) and without the goal reward (red line). The fifth training curve (purple line) includes all rewards,

as a comparison. Whilst this figure shows the relative importance of the different reward types, it is important to realise that all four types of reward were included in the virtual environment.

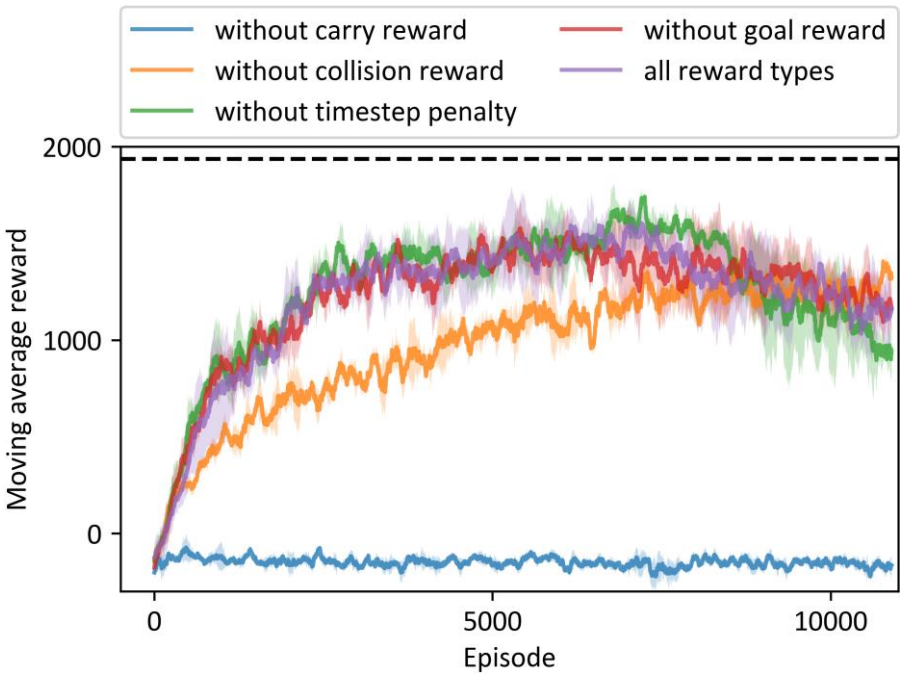

Figure 6-8 Training curves of the reinforcement learning agent in the virtual environment, where one type of reward is withheld from the reinforcement learning agent, namely without carry reward (blue line), without collision reward (yellow line), without time step penalty (green line), without goal reward (red line). An additional training curve is provided for comparison, where all types of reward are granted (purple line).

It can be seen from the figure that the most important type of reward is the carry reward, as seen by the fact that disabling this reward component completely prevents learning by the reinforcement learning agent. Without the presence of the carry reward, the only motivating force that incentivises the reinforcement learning agent to steer a laser-trapped microparticle to a target position is the goal reward. However, the goal reward is a 'discrete' reward that can only be obtained at best once in an episode, and thus relying solely on this reward to reinforce the behaviour of translating a laser-trapped microparticle to the target position (initially through random exploration) is highly inefficient, and is evidently insufficient for the chosen reinforcement learning algorithm (i.e., TD3). It is also evident, in Figure 6-8, that disabling the acquisition of the goal reward (red line) does not have a significant impact on the learning of the reinforcement learning agent. The collision reward (yellow line) is also a type of reward that has an observable influence on the learning of the reinforcement learning agent. It can be seen from the figure that, whilst eventually the training curve (i.e., yellow line) plateaus at approximately the same level as the training curve with all types of reward enabled (i.e., the purple line), the speed at which the

yellow line converges is much slower than that of the purple line. This can be explained by the fact that disabling the collision reward had a very limited effect on the path-planning ability of the reinforcement learning agent to steer a laser-trapped microparticle to the predefined target position but had an obvious influence over the collision avoidance ability of the reinforcement learning agent. However, owing to the fact that translating the laser-trapped microparticle to the target position yields a much higher discounted episodic reward than the penalty for colliding with a free-moving microparticle, collision avoidance can eventually be learnt in a later stage of the training, even when the collision penalty is absent. Despite this, when considered in isolation, the goal reward, and the time step penalty each seem to have a very limited impact on the learning of the reinforcement learning agent (especially during the early stages of training). Many factors could give rise to their apparent insignificance, such as their relative magnitudes with respect to other types of reward, or perhaps they are only effective in combination with other types of reward. Investigating these multifactorial effects is beyond the scope of this study, and therefore for the remainder of this chapter, all four types of reward were included in the training of the reinforcement learning agents.

6.5.2 Discount Factor

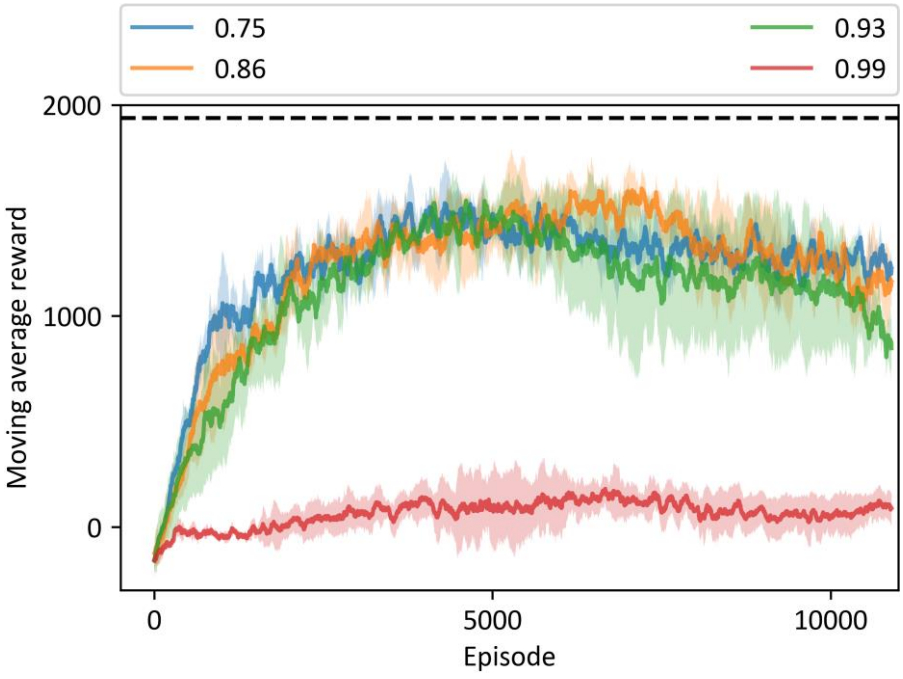

Figure 6-9 Training curves of the reinforcement learning agent with different discount factors, namely 0.75 (blue line), 0.86 (yellow line), 0.93 (green line) and 0.99 (red line).

Another factor that can have a significant effect on the learning of a reinforcement learning agent is the discount factor γ. Figure 6-9 shows the effects of different discount factors in training a reinforcement learning agent in the virtual environment. The discount factor γ is usually used in

reinforcement learning to exponentially weight a piece of future reward solely based on its distance with respect to the present (count in time steps). This practice is usually employed to mathematically add a bounding property to a reinforcement learning problem, but it also modifies the decision making of a reinforcement learning agent to prioritise short-term decisions over long-term decisions (i.e., a reward in the far future is less valuable than a numerically equal reward obtained in a near future, due to exponential weighting). A discount factor just slightly below unity is usually adopted, in order for a reinforcement learning agent to be incentivised to perform long-term decision-making (commonly adopted values include 0.99 and 0.997). In both the physical and virtual environment of the optical tweezers setup, the limited field of view imposes restrictions on the area which a reinforcement learning agent is able to observe at each time step. This restriction means that free-moving microparticles that may in future intercept the path of the laser-trapped microparticle, may at the present time not yet be visible to the decision-making reinforcement learning agent. Hence the ability for the decision-making agent to deduce appropriate actions for the far future is directly compromised by this imposed visual restriction. This is a classic case of a Partially-Observable Markov Decision Process (POMDP), and much literature has covered this area (Jaakkola et al., 1994). In this study, the partial observability is addressed via the discount factor. Recall equation 3-6, let R be the sum of a series of pieces of future reward r_t, each of which is exponentially weighted by a discount factor γ based on its step count t with respect to the first piece of reward

$$R = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \gamma^{n-1} r_n = \sum_{t=1}^{N=n} \gamma^{t-1} r_t$$

Whilst the precise r_t received is critical in training the reinforcement learning agent, here we focus on the bounding limit imposed by the discount factor γ and so r_t can be tentatively removed from the summation. Therefore, a geometric series Y can be written here to describe the bounding property of R

$$Y = 1 + \gamma + \gamma^2 + \cdots \gamma^{n-1} \qquad (6-2)$$

In addition, assuming that the length of this series is infinite $n \to \infty$, the limiting sum of this series Y can be given by

$$\lim_{n \to \infty} Y = Y_\infty = \frac{1}{1-\gamma}$$

If the partial observability means that the decision-making agent has a limited ability to decide actions that will occur more than n steps into the future, it can be regarded as that it is baseless for the agent to decide any action after n steps. Therefore, future rewards that the agent would obtain

for actions that are more than n steps into the future should be omitted in the total discounted reward. The sum of the first n items in the series Y can be written as

$$Y_n = \frac{1-\gamma^n}{1-\gamma}$$

In this case, in order for a decision-making agent to deduce future actions sensibly, Y_n should take up a great proportion of Y_∞. A hypothetical 99 % can be assumed here, and Y_n and Y_∞ thus can be written together as

$$Y_n = \frac{1-\gamma^n}{1-\gamma} = 0.99 Y_\infty = 0.99 \frac{1}{1-\gamma} \qquad (6-3)$$

It can be simplified to $\gamma_n = \sqrt[n]{0.01}$. Considering the maximum allowed velocity (0.015 mm/s), the time interval between two adjacent steps (0.1 seconds), and the shortest distance between the centre of the observation (where the laser-trapped microparticle located) and the boundary of the field of view (dimensions of the field of view are 0.094 mm by 0.1185 mm, so this distance is 0.094/2=0.047 mm), a decision agent can deduce at least 32 sensible consecutive future actions ($\left.0.047\ (mm)\right/ 0.015\ (mm \cdot s^{-1}) \cdot 0.1(s) \sim 31.3$), based on a single observation that is constrained by the given field of view. Based on this value, $\gamma_{32} = \sqrt[32]{0.01} \sim 0.86$. The orange training curve in Figure 6-9 shows the learning capability of the reinforcement agent with a discount factor of 0.86. For comparison, the training curve for discount factor $\gamma_{16} = \sqrt[16]{0.01} \sim 0.75$ is shown by the blue curve, $\gamma_{64} = \sqrt[64]{0.01} \sim 0.93$ by the green curve and the commonly adopted discount factor value of 0.99, $\gamma_{458} = \sqrt[458]{0.01} \sim 0.99$ is shown by the red curve (all tested under the same training parameters other than discount factor). The discount factor of 0.86 (theoretically justified on the basis of the extent of partial observability) yields the highest peak performance, suggesting that the proposed method for estimating an optimal discount factor is valid for this environment with a limited field of view. Essentially, the proposed method suggests that planning future actions beyond the extent of the current field of view may be fundamentally meaningless, and therefore the degree of future planning should be limited by the extent of the field of view through adjusting the value of the discount factor. It must be clearly stated here that the proposed method for estimating the discount factor is, by no means, rigorous. The problems with the proposed method are twofold:

1) In equation 6-2, omitting pieces of reward r_t in R can be critical if r_t varies significantly.
2) In equation 6-3, the estimate of $\gamma$for a finite number of steps (Y_n), approaches 0 if, instead of 99 %, we attempt to include a far greater fraction of Y_∞ within the estimate, e.g., $\sqrt[32]{10^{-40}} \sim 0.06$.

In addition, as mentioned previously, other literature has discussed methods that address the POMDP in other ways.

6.6 Application in the Virtual Environment

The reinforcement learning agent, after training, was first evaluated in the virtual environment. Figure 6-10 shows an example of a trajectory, where the laser-trapped microparticle and its associated trajectory are colour-coded in red, other free-moving microparticles and their respective trajectories are colour-coded in green, and the predefined target position is shown as a blue hollow circle. The entire trajectory of the laser-trapped microparticle and the entire trajectories of other free-moving microparticles are shown in Figure 6-10 a). The blurred appearance of the green circles corresponds to the movement of each free-moving particle during the episode. Likewise, the gradient colour of the red line shows the path of the laser-trapped particle during the episode. The extent of the initial field of view is shown in Figure 6-10 b), colour-coded as a white rectangle, within which the initial distribution of the observable free-moving microparticles is shown. The white square therefore shows the complete camera information available to the reinforcement learning agent at the start of the episode. Figure 6-10 c) shows the state at the end of this episode, when the laser-trapped target pattern eventually reaches the predefined target position without having collided with any of the free-moving microparticles. It is clear that the reinforcement learning agent, acting in the virtual environment, does not take a direct route to the target position (blue circle), due to the requirement to avoid other particles.

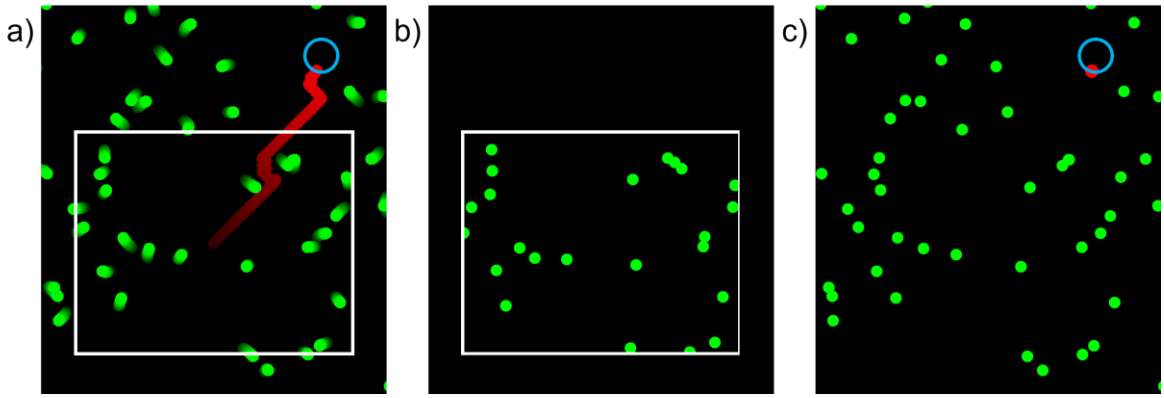

Figure 6-10 Simulation results of the reinforcement learning agent in the virtual environment, showing a) the entire trajectory, b) extent of the field of view, and c) the state at the end of the episode.

A more detailed demonstration is shown in Figure 6-11, where 12 target positions (shown as blue hollow circles in the figure) with 30-degree separations are positioned on a circle with a radius of 0.093 mm. For each target position the laser-trapped microparticle is initialised at the centre of the circle. By controlling the seeding (i.e., the initial velocities and the initial accelerations of the

free-moving microparticles) and the random accelerations applied at each step to the microparticles, the 12 trajectories were generated in the same virtual environment. In other words, the figure shows the result of testing the same environmental conditions (with the free-moving microparticles having the same trajectories for each rollout) for 12 different target positions. Figure 6-11 a) and b) shows two examples of this demonstration, where, in each case, the 12 trajectories of the laser-trapped microparticles are colour-coded in red, the trajectories of other free-moving microparticles are colour-coded in green, and the initial field of view of the observation is colour-coded as a white square. In some positions, whilst it appears that the red and green trajectories intersect, implying a collision, these circumstances correspond to two particles being present at the same position but at different times. Whilst the laser-trapped microparticle is initially placed at the centre of the circle, ensuring that the distances between the laser-trapped microparticle and the target positions are identical, the total number of steps for the reinforcement learning agent to reach each target position differed. The free-moving microparticles are therefore rendered up to the step count of the longest trajectory. It can be seen from Figure 6-11 that the reinforcement learning agent prefers to move diagonally at the start of each episode. This can be justified by the fact that, in order to simulate the physical environment, the velocities along the X and Y axes are limited independently, and thus the maximum translation speed is only achievable when moving diagonally. This is an interesting example of an unexpected strategy devised by a reinforcement learning agent, when given the task of maximising a set of reward conditions.

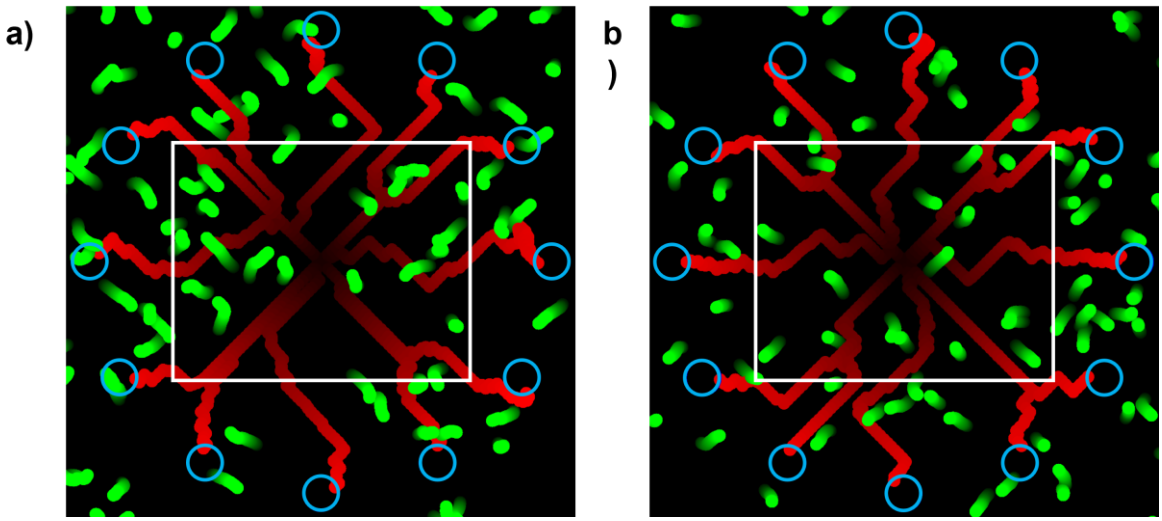

Figure 6-11 Two example simulation results of the reinforcement learning agent in the virtual environment, where predefined targets are located at 12 different positions, 0.093 mm away from the initial position of the laser-trapped microparticle. The movements of the free-moving microparticles are randomly generated but were kept the same for each episode (with one target selected per episode).

6.7 Application in the Physical Environment

The trained reinforcement learning agent was then applied to the physical environment. No further training was completed for the reinforcement learning agent when it was transferred from the virtual environment to the physical environment. Four trajectories of the laser-trapped microparticles that were controlled by the reinforcement learning agent in the physical environment are shown in Figure 6-12. Each subfigure in Figure 6-12 shows the path of its respective experimental results, which has been created by stitching together unprocessed camera observations of the laser-trapped microparticle (i.e., the sensory input) that were recorded during the episode. The predefined target position on each subfigure is shown as a green hollow circle. Note that, in the experiments that were made in the physical environment, the target positions were not necessarily set at 0.093 mm away from the initial positions of the laser-trapped microparticles (unlike in Section 6.6, where the target positions are always 0.093 mm away from the initial position of the laser-trapped microparticles), and therefore, the stitched panoramas of the experimental results differ in their sizes. In order to fit these panoramas of experimental results into the same figure, different degrees of resizing are applied to the panoramas, and therefore each panorama in Figure 6-12 is given a dedicated scale bar.

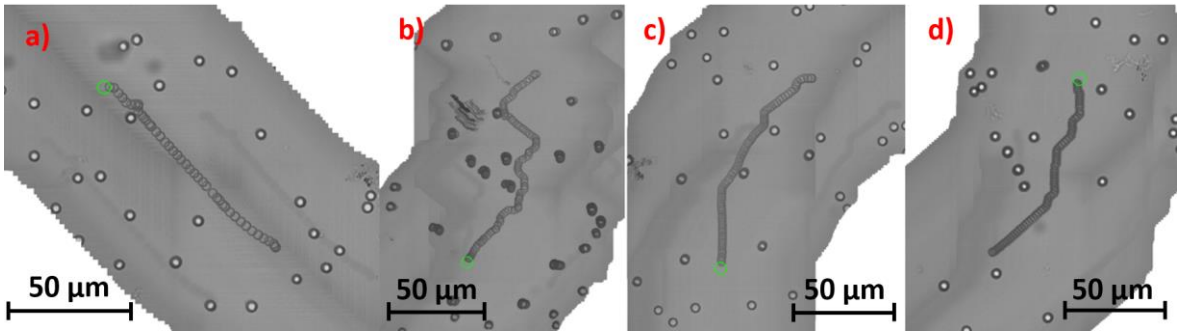

Figure 6-12 Panoramas of the trajectories of the real-world laser-trapped microparticles that were controlled by the reinforcement learning agent in the physical environment.

The trajectory from Figure 6-12 d) is further detailed in Figure 6-13, which depicts a) the first, b) the middle and c) the last camera observations of the trajectory in light grey. In the figure, the previously visited but not currently visible areas are shaded in deeper grey, whilst the predefined target position is shown as a green hollow circle. In each subfigure of Figure 6-13, the action determined by the reinforcement learning agent is shown a red arrow. Although the reinforcement learning agent only determines the velocities along the X and Y axes at which the laser-trapped microparticle should move in the next fixed period of time, these two velocities can be combined into a resultant vector, and therefore can be shown as an arrow. The green arrows on the same figure are the vectorised displacements (i.e., the positional inputs, pointing from the laser-trapped

microparticle to the predefined target pattern) which the reinforcement learning agent received (i.e., as the directional input) at the respective time steps. For clarity, the previously visited but not currently visible areas shaded in deep grey are only shown in the figure to present the past trajectory for the human reader. The information contained in these areas was not inputted to the reinforcement learning agent, which only had access to the current processed camera image (with its restricted field of view) and the directional vector corresponding to the target position.

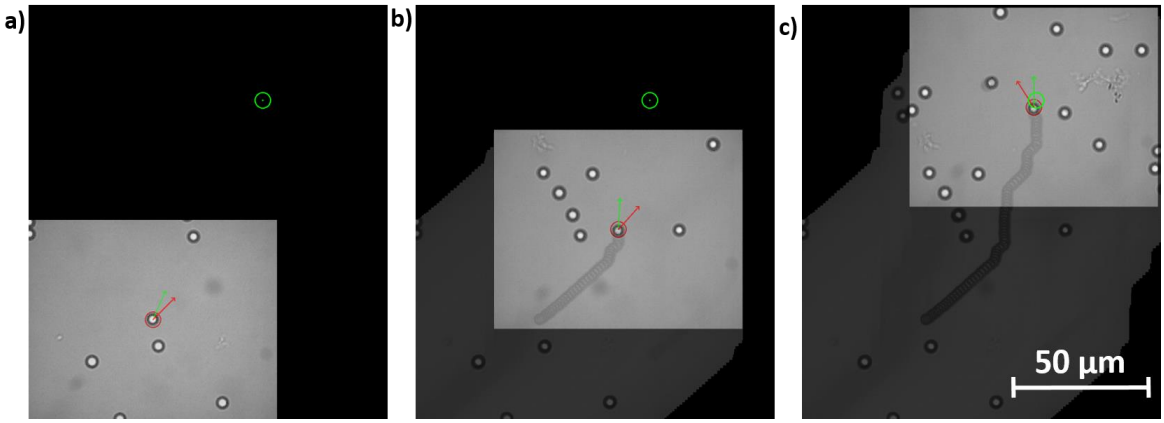

Figure 6-13 Panoramas of a) the first, b) the middle and c) the last time step of the trajectory of a real-world, laser-trapped microparticle, that was controlled by the reinforcement learning agent in the physical environment. The red arrow shows the action decided by the reinforcement learning agent and the green arrow shows the positional inputs. Note that the green arrow is not pointing exactly at the target position due to backlash and single-channel communication between the motorised stages and the decision-making hardware.

As previously stated in Section 0, in the physical environment, two motorised linear stages were used independently to translate the sample holder in the directions along the X and Y axes. Therefore, the laser-trapped microparticle is translated with the highest velocity when both the X and Y translation stages are moving at their maximum speed. In other words, the laser-trapped microparticle can be translated at its maximum speed only in the diagonal directions. Because of this, even for movements in which only one linear stage is required (i.e., along the X or Y axis), moving the other linear stage does not reduce the speed at which the laser-trapped microparticle is translated in the desired direction. Therefore, the reinforcement learning agent is frequently observed to move in the direction along one stage axis (i.e., the X or Y axis), whilst also alternating its velocity on the perpendicular stage axis between the maximum positive and the maximum negative speeds. This pattern of movements forms a zig-zag trajectory for the laser-trapped microparticle, which can be observed in both Figure 6-11 and Figure 6-12 (most notably in Figure 6-12 b)).

6.8 Application in a Hybrid Environment

Whilst the reinforcement learning agent is trained in the virtual environment, it can be seamlessly adopted in the physical environment, as these two environments are treated equally in terms of input and output by the reinforcement learning agent. It is therefore also possible to construct an environment in which the input is a hybrid of both the virtual and physical environments, and in which the output is used to control a real laser-trapped microparticle in the physical environment. Here, we refer to this combination of environments as a 'hybrid' environment. Potential applications here include the computerised addition of constraints in a physical environment, for example limitations on stage movement.

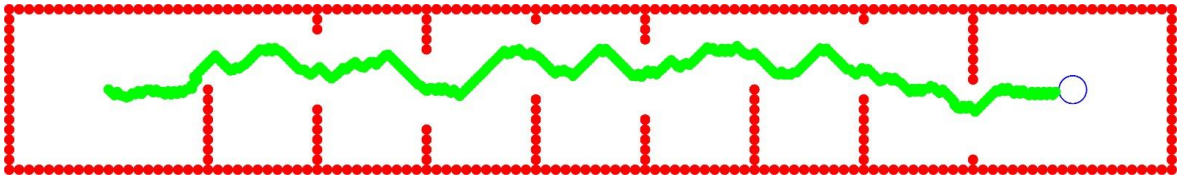

Figure 6-14 Chosen path of the reinforcement learning agent through a 'maze' created in the virtual environment.

A special 'maze' virtual environment was built, in which the fictional microparticles are immovable, and are used to form the boundaries of the maze. This is an elegant approach, as the reinforcement learning agent is already aware of the requirement of avoiding collisions with the surrounding microparticles. In this maze environment, the distribution of the fictional microparticles is no longer governed by a probability distribution. Instead, a maze shape is constructed by arranging fictional microparticles to form an enclosed rectangle that has several vertical barricades (i.e., gate-shaped obstacles) in between the two horizontal sides of the rectangle. Figure 6-14 shows the trajectory of a fictional laser-trapped microparticle (colour-coded in green), which is moved by the reinforcement learning from one side of the maze to the opposite side of the maze (the target position is shown as a blue hollow circle). The agent is able to guide the microparticle past the vertical barricades without colliding into any of the immovable, fictional microparticles (shown as red solid circles in the figure).

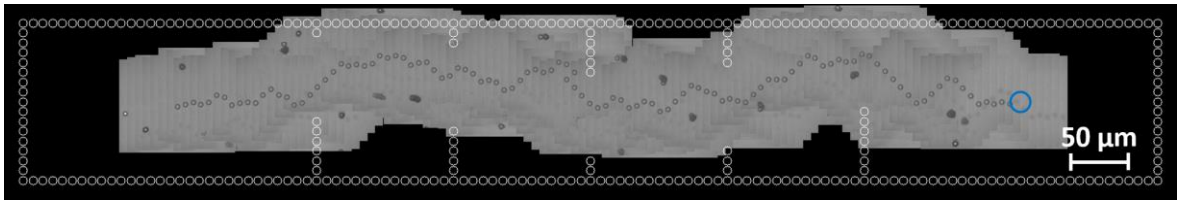

Figure 6-15 Panorama of the trajectory of the real-world microparticle that was controlled by the reinforcement learning agent in the hybrid environment.

A real-world experiment was then conducted by creating a hybrid environment where observations of the virtual 'maze' environment were overlaid onto observations of the real-world, physical environment. The reinforcement learning agent therefore was provided with an input image that corresponded to the combination of the real-world particles (which were moving) and the virtual maze particles (which were static). Figure 6-15 shows the trajectory of a real-world laser-trapped microparticle which was translated by the reinforcement learning agent from one side of the maze to the opposite side of the maze in the hybrid environment (the target position is shown in a blue hollow circle), where there existed both real-world free-moving microparticles and fictional immovable microparticles. In this hybrid environment, the processed sensory input renders the fictional immovable microparticles (from the virtual part of the environment) identically (as solid circles) to the real-world, free-moving microparticles (from the physical part of the environment). However, for visual clarity, in Figure 6-15 the fictional microparticles are shown as white hollow circles. As shown in the figure, the reinforcement learning agent was able to maneuverer the laser-trapped microparticle to the target position, while also avoiding collisions with both the real-world free-moving microparticles and the virtual obstacles that were made of fictional immovable microparticles. As the reinforcement learning agent was trained in a virtual environment where the distribution of free-moving microparticles was of a random nature, it was almost impossible for the reinforcement learning agent to encounter gate-shaped obstacles composed of fictional microparticles during training. The fact that the reinforcement learning agent was successful in navigating this hybrid environment provides strong evidence that agent has indeed learnt the general principles of both obstacle avoidance and path-planning. This demonstrated technique has as a range of potential applications, such as the potential to be incorporated into a microfabrication process as a means to transport parts from one position to a different position on a workpiece, where barricades composed of fictional microparticles can be used as a means to prevent the part that is being moved from visiting certain areas on the workpiece.

6.9 Conclusion

In conclusion, reinforcement learning has been applied to the field of optical tweezers, in order to enable the capability for automated laser-based non-contact translation of single particles from position A to position B, whilst avoiding collisions with other particles. The reinforcement learning agent was trained in a virtual environment, and applied to a real-world physical environment, where no additional training was required in the physical environment. Through an elegant technique, an additional layer of information (corresponding to a maze structure) was overlaid onto

the physical environment, in order to create a ‘hybrid’ environment which the trained reinforcement learning agent was also successful in navigating.

Chapter 7 Tool-Path Generation for Femtosecond Laser Machining

Femtosecond laser machining offers the capability for extremely precise material removal, with sub-micron scale resolution possible. Typically, a single focal spot is used for machining, and hence, to machine a larger scale structure, the laser focus needs to be moved relative to the sample. This generally involves moving the sample via translation stages over a predetermined set of coordinates. For more complex shapes, the optimal path for translation can be challenging to identify. In this chapter, it is shown how reinforcement learning can be used to determine an effective set of coordinates for controlled firing of laser pulses, in order to laser machine specific shapes.

For the work in this chapter, the author created, optimised and trained the reinforcement learning agents, collected all laser machining experimental data, and performed all analysis. The laser machining experimental setup, and its associated automation, was constructed previously by Dr Benjamin Mills. The virtual environment was constructed by the author with the help of Dr Matthew Praeger.

7.1 Motivation

In a diverse range of laser materials processing techniques, laser energy is directed onto a target material in order for a chemical or physical change to occur. For instance, in laser machining, the laser energy is used to remove part of the target material via the laser ablation effect. In this case of laser welding, the laser energy is used to joint two pieces of adjacent or stacked target materials through creation of a melt-pool. In laser chemical vapour deposition, the laser energy is used to increase local temperature on a target surface to cause vapour production. These applications, and many other laser-related materials processing techniques, all involve, maybe to a different extent, precise control over the position on the target material at which laser energy is applied. In many applications, the level of precision required for the intended laser-induced effects on the target material cannot be manually achieved, and therefore automated computer-controlled hardware is generally required. Many studies have been conducted over the past few decades to address this beam positioning challenge, and a variety of methods have been proposed across different fields in laser materials processing. For instance, beam position can be controlled by moving the sample holder or beam delivery optics via motorised stages, through changing the beam path via galvanometer scanning mirrors in conjunction with an f-theta lens, through modifying the spatial

intensity profile of the beam via a spatial light modulator (e.g., a digital micromirror device or, liquid-crystal-based spatial light modulator). In practice, multiple methods can be employed sequentially or simultaneously to achieve the required high precision beam positioning. Whilst the underlying physics behind the light-matter interactions and the configurations of the beam position controls can differ, a common challenge in precise beam positioning is to choose a set of positions on a workpiece at which laser energy should be applied, or trajectories which movable tools should follow, to achieve the intended result for the application.

In this chapter, a laser machining system is demonstrated, in which reinforcement learning is used to precisely control the beam positioning through translation of the sample holder with respect to the laser focus, via two motorised stages (i.e., X and Y axes). The reinforcement learning agent is therefore able to choose positions on a workpiece to which laser the pulses are incident in order to machine a predefined target pattern. At discrete time steps, an observation of the workpiece is made via a camera, and the subsequent position to apply a laser pulse is determined by the reinforcement learning agent, calculated using both the current camera image of the sample and the predefined target pattern. This laser machining system therefore enables real-time control of sample translation and pulse-firing to laser machine a pattern on a target workpiece in accordance with a predefined target pattern. In addition, as the sample is continuously observed during the laser machining process, the laser machining system has the capability to detect (and correct for) any erroneously positioned laser pulses, in real time, where this correction process is automatically achieved through the re-calculation of subsequent sample translations.

In this experiment, a virtual environment (see Section 3.4) was used to train the reinforcement learning agent, where the virtual environment is a simulated approximation to the real-world laser machining environment. The training of a reinforcement learning agent in a virtual environment minimises the reliance on real-world data collection and prevents experimental apparatus from being damaged by the exploratory behaviours of the reinforcement learning agent. A significant challenge is of course making sure that the virtual environment is a suitable replica of the real-world physical environment; as the objective here is to train the reinforcement learning agent in the virtual environment, and then apply the trained agent in the real-world experimental setup. To achieve this, manually constructed computer vision algorithms were used to extract key elements from the camera observations, with regards to the positions of laser-machined features. The visual appearance of this extracted information was simulated directly in the virtual environment, enabling a reinforcement learning agent to be applied interchangeably between the virtual and the physical environments.

7.2 Experimental Setup

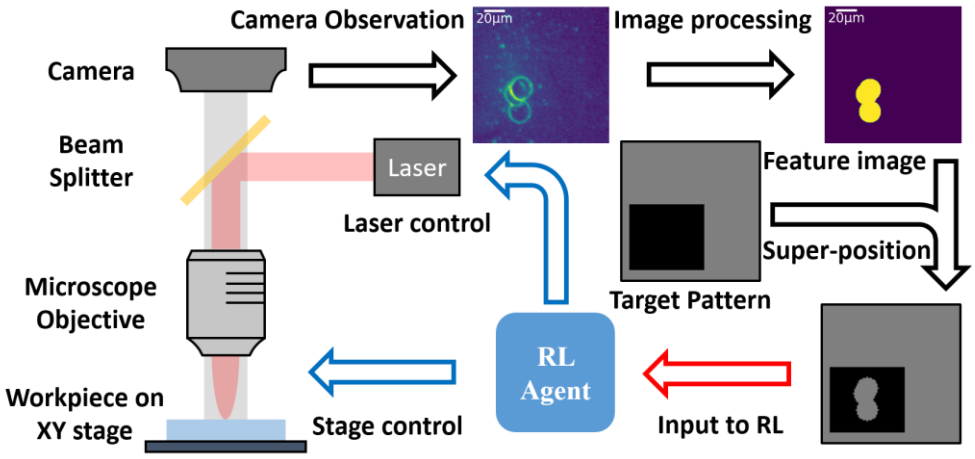

Figure 7-1 Schematic of the experimental setup.

The experimental setup is schematically depicted in Figure 7-1, in which laser pulses (190 fs, 1025 nm, 1 mJ) from a Light Conversion Pharos-SP-1mJ laser system were first reflected by a beam splitter, and then imaged onto a workpiece via a microscope objective (Nikon 20×, N.A. 0.4). In order to enable simultaneous observations of laser-machined structures on the workpiece, a CMOS camera (Thorlabs DCC1645C) was positioned above the beam splitter. The workpiece was a 1 mm thick, fused silica, microscope slide which was machined with a fluence of ~4.6 J/cm^2. Various studies have been conducted verifying that machined structured can be observed with a machining fluence of around 5 J/cm^2 on fused silica (Chimier et al., 2011). An XYZ motorised translation stage (formed of three Thorlabs LNR502/M stepper motor actuated stages) was used to hold the workpiece. The Z motorised stage moved along the direction of the incident pulses, and hence was used only for manually adjusting the vertical position of the workpiece, with reference to the microscopic objective lens, in order for laser pulses to be focused onto the front surface of the workpiece at the start of the experiment. The XY translation stages which were orthogonal to the direction of incident pulses, and to each other, were used during the experiment at discrete time steps to control the position on the workpiece at which laser pulses were incident. The spatial intensity profile of the laser pulses was of Gaussian shape, which laser machined a visible structure that was of circular shape with a diameter of approximately 18 µm ± 2 µm. The variation in size of the machined structure was understood to be a consequence of micron-scale changes in the focal position when the X and Y stages were translated, hence changing the size of the focussed beam on the sample.

7.3 Neural Network Architecture

The connectivity of the neural networks that acted as function approximators in the Proximal Policy Optimisation algorithm (PPO) is shown in Table 7-1, which was similar to the connectivity of the neural networks that was described in Section 6.3. Unlike in the TD3 algorithm where the critic determined state-action values from state-action pairs, the critic of PPO determines the value of a state from the state observation (i.e., $V(s_t)$). This means that the exact same configuration of the neural network could be used for both the actor (i.e., $\pi(a|s_t)$) and the critic. Additionally, in the PPO algorithm, both the actor and the critic utilise the same neural network as a feature extractor, to map a sensory input to an abstract representation of this sensory input. This is commonly referred to as the 'parameter sharing', which means that the actor and critic effectively utilised the same set of parameters inside a neural network to determine actions and values of states. To that end, the same set of parameters were also tuned towards the directions of minimising both the policy gradient loss and the value function loss (a hyperparameter is usually multiplied to the value of function loss in order to reduce the variance of the total losses). The parameter sharing is implied in Table 7-1 by the fact that there is no horizontal line separating the actor and critic — both the actor and the critic were connected to the first fully connected layer. Since, in this work, the actor controlled two orthogonal translation stages, the output of the actor was of the shape 1 × 2, whereas the critic only determined the values of states, and therefore its output was of the shape 1 × 1. Other parameters can be found in Table A 4

Layer name	Input size (pixels)	Parameter (pixels)	Output size (pixels)
Input layer	100 × 100 ×1	-	100 × 100 × 1
Conv_1	100 × 100 × 1	8 × 8 × 32, stride of 4	25 × 25 × 32
Conv_2	25 × 25 × 32	4 × 4× 64, stride of 2	13 × 13 × 64
Conv_3	13 × 13 × 64	3 × 3 × 64	13 × 13 × 64
Flatten	13 × 13 × 64	-	1 × 10816
Fully_Connected_1	1 × 10816	10816 × 512	1 × 512
Actor Critic	1 × 512 1 × 512	512 × 2 512 × 1	1 × 2 1 × 1

Table 7-1 Neural network architecture used for both the actor and the critic with the PPO algorithm.

7.4 The Physical and Virtual Environment

The flow diagram of the experiment rollout in the physical environment is shown in Figure 7-1. The experiment in the physical environment was operated in a cycle of repeating steps, which was halted only when certain termination conditions were met. Each repeating step was composed of the following sub-steps:

1) At the start of an experiment, the X and Y axes of the motorised stage are moved to a fresh area on the workpiece (i.e., an area without any previously laser-machined structures). At the start of each step (i.e., the first sub-step), a camera observation of the workpiece is made. This stage coordinate is marked as the 'imaging coordinate', and this camera observation is called the 'before pulse' observation for this step.
2) If the current step is the first step in the experiment, a predefined target pattern is shown to the reinforcement learning agent. If the current step is not the first step in the experiment, a superposition of the target pattern and a binary 'feature image' is sent to the reinforcement learning agent instead. The agent then deduces an action based on the target pattern or superposed input. Here, the 'feature image' is a processed camera observation of the workpiece in which the laser-machined structures and the surrounding material are rendered in two different colours. The deduced action contains two numerical values which correspond to the XY coordinates on the workpiece at which the next laser pulse should be applied.
3) The X and Y motorised stages are moved to the XY coordinates deduced by the reinforcement learning agent. A laser pulse is thereafter applied to the workpiece once the stages have completed their movement.
4) The X and Y motorised stages are returned to the 'imaging coordinate' (i.e., starting position), and a camera observation of the workpiece is made. This camera observation is called the 'after pulse' observation for this step.
5) A series of image processing techniques are applied to the 'before pulse' and 'after pulse' observations of the workpiece, that were obtained in the current step. This processing allows the position of the newly laser-machined structures, made in the current step, to be identified. This information is used to update the binary 'feature image'.

In summary, for each cycle, a camera image is taken of the current sample at the starting position ('before pulse'), which is processed by the reinforcement learning agent, the sample is moved to the new position and a laser pulse is fired, the sample is moved back to the starting position and another camera image is taken ('after pulse'). This procedure was used, so that the 'before pulse' and 'after pulse' images of the sample were both recorded at the same XY position,

which made comparison of the images considerably easier. In practice, the 'after pulse' observation of the N^{th} incident pulse is identical to the 'before pulse' observation of the $N+1^{th}$ pulse, therefore, in order to reduce processing time per step, duplication of the same camera observation was not made; instead, the 'after pulse' observation of the N^{th} incident pulse was used directly as the 'before pulse' observation of the $N+1^{th}$ pulse. This cycle of repeating steps was only halted when one of the following termination conditions was met:

1) The shape of the target pattern had been completely laser machined on the workpiece (i.e., good exit condition).
2) A pulse had been applied at a position on the workpiece that does not contribute to the completion of the laser machining of the shape of the target pattern. This includes applying a pulse at the surrounding material on the workpiece, or areas on the workpiece where previously laser-machined structures already existed (i.e., bad exit condition).

An artificial example of a superposed input to the reinforcement learning agent (i.e., the superposition of a predefined target pattern and a 'feature image') is shown in Figure 7-2 a). This input provides critical information to the reinforcement learning agent, namely the areas on the workpiece that should and should not be machined. In this example, the target pattern is of square shape, and seven pulses have been applied to the workpiece crossing it diagonally, with a fixed spatial spacing between consecutive incident pulses across the target pattern. Since the input to the reinforcement learning agent is a binary image, only two values are permitted to exist on the input image, namely 'background' value $c^{background}$ (abbreviate to c^b for the remainder of this chapter) and 'remaining target' value c^{target} (abbreviate to c^t for the remainder of this chapter). The 'background' value c^b, which is shown in light grey in the figure, is used for areas on the workpiece at which laser pulses should not be applied and/or have already been applied. On the other hand, the 'remaining target' value c^t, which is shown in deep grey, is used for areas on the workpiece at which laser pulses should be applied. In this simple model, applying a pulse to pixels in the 'remaining target' area causes them to be converted from 'remaining target' to 'background' status. This is achieved by changing the values within the detected area of laser-machining from c^t to c^b. Notably, in the case where a laser-machined structure is located at the boundary between a 'remaining target' area and a 'background' area only the region of 'remaining target' that is overlapped by the laser-machined structure changes its value to c^b. Regions of the 'background' that are overlapped by the laser-machined structure simply remain as 'background', retaining their value c^b. Figure 7-2 b) shows a different visualisation of this machining process, which contains extra information about the laser machining process, such as the initial target pattern and the order of the incident pulses. This information is not accessible to the reinforcement learning but is nevertheless helpful for human readers to interpret the rollout of a laser machining experiment

(and thus this style of visualization is used in the remainder of this chapter). In Figure 7-2 b), the initial predefined target pattern is shaded in yellow (the target pattern is still a square-shaped pattern), the positions at which laser pulses are applied are marked as red dots, a colour-coded line is used to connect the red dots with its colour indicating the order of the incident laser pulses, finally, the green contour shows the merged shape of the laser-machined structures. This figure shows laser machining with seven laser pulses, over a size scale of approximately 40 microns.

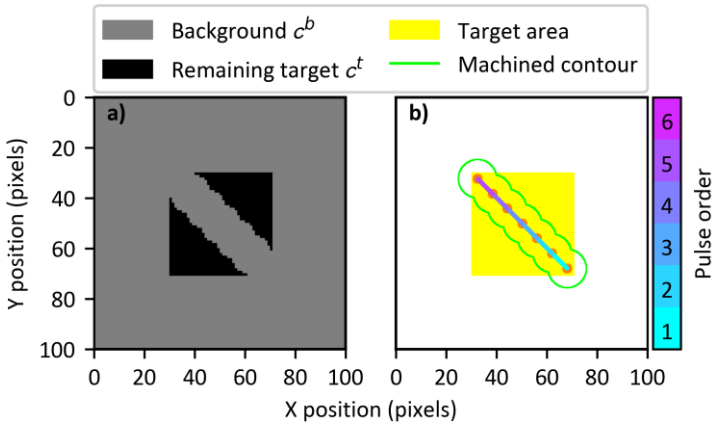

Figure 7-2 An artificially made superposition of the initial target pattern and detected laser machining features, processed to segment the workpiece into 'background' and 'remaining target' regions, presented in the form of a binary image. b) An alternative visualisation of the same machining trajectory as a) which is intended for human viewers and clearly shows both the initial target area and the order of the incident laser pulses.

As stated previously, the input to a reinforcement learning agent is a binary image which is a superposition of a predefined target pattern and a 'feature image' (as shown in Figure 7-1). This 'feature image' is a two-dimensional array that segments the areas of laser-machined structures from the background material by processing a pair of images collected before and after an incident pulse. The image processing procedure, which has the purpose of identification of the most recent laser machined structures, consists of the following steps:

1) Image subtraction of the 'before pulse' and 'after pulse' camera images of the workpiece.
2) Smoothing and de-noising of the subtracted image, by applying Gaussian filtering.
3) Adaptive image thresholding in order to extract features of the laser-machined structure.
4) Image erosion to localise the extracted features and remove small items such as debris.
5) Hough transform to identify the coordinates of the circular shaped feature corresponding to the laser-machined structure.

A depiction of this image processing procedure is shown in Figure 7-3 as a flow diagram. The image manipulation methods involved are all commonly used in computer vision studies

(e.g., object recognition, image morphology), therefore, these algorithms are generally well-optimised, and are available in many libraries across many operating systems. In this work, built-in functions from the *Python* version of the *OpenCV* library (Bradski and Kaehler, 2000) were used. The time taken for processing a pair of images (i.e., a 'before pulse' image and a 'after pulse' image) was approximately 0.15 seconds, without any optimisation such as parallel computing or code optimisation, in a low-end desktop configured with a 4th generation Intel i5 CPU. At each step, by applying this image processing procedure, the relative position of the most recent laser-machined structure could be determined, with respect to the 'imaging coordinate'. This piece of information, in combination with relative positions of previously laser-machined structures, can be used to produce a 'feature image'. This feature image segments a camera observation into two areas, namely the area where laser-machined structures are located and the area at which incident pulses have not been applied. Superposition of this feature image with the predefined target pattern produces a binary input image that provides the required information for the reinforcement learning agent to determine the XY coordinate on the workpiece at which the next laser pulse should be applied. Compared to monitoring a laser machining process by only tracking the movements of linear stages along X and Y axes, this image-based approach directly observes the workpiece and the laser-machined structures on the workpiece in real time. This enables the ability of the reinforcement learning agent to adjust its tool-path design, in real-time, to compensate for any machining errors that arises from an inadvertently missing or incorrectly positioned incident pulses.

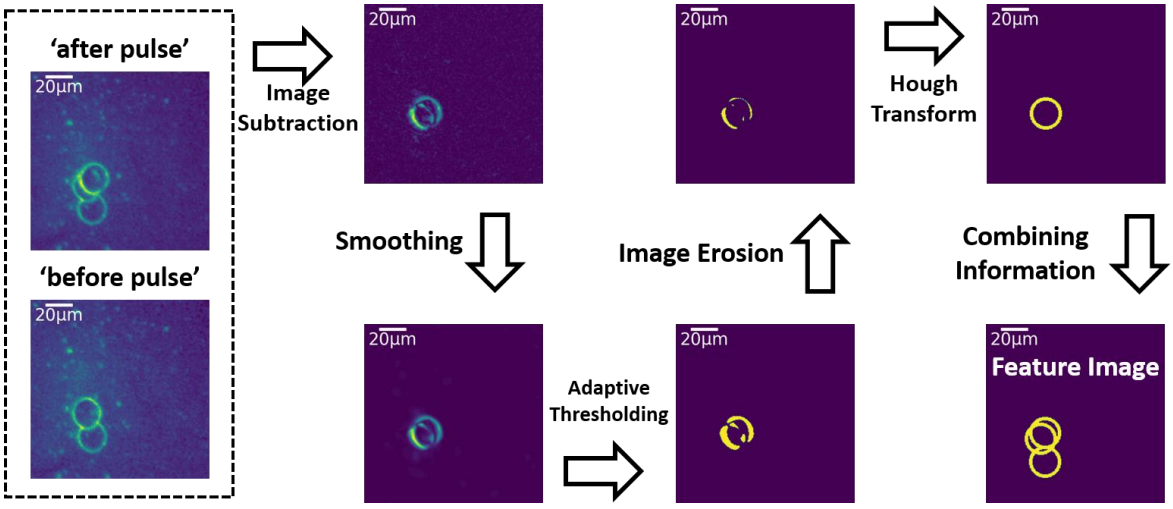

Figure 7-3 Flow diagram of the image processing procedure. For visual clarity, three hollow circular shapes are rendered in the final feature image in order to match the visuals of the laser-machined structures on the camera observation. In practice, both in the physical and virtual environments, solid circles were rendered instead.

The virtual environment is schematically depicted in Figure 7-4. A reinforcement learning agent interacts with the virtual environment in the same way that it interacts with the physical environment. This is intentional, as the purpose of the virtual environment is to allow a reinforcement learning agent to develop a certain skill without the need to collect data on the physical environment. In the case of this study, the task is selecting appropriate XY coordinates for laser machining, in accordance with segmented observations of a workpiece and a predefined target pattern. The virtual environment operates in a similar manner to the physical environment, with actions being determined by the reinforcement learning agent at each discrete time step. In the virtual environment, each step is composed of the following sub-steps:

1) A segmented observation of a virtual workpiece (i.e., a feature image) is made.
2) If the current step is the first step, the target pattern is input to the reinforcement learning agent. If the current step is not the first step, a superposition of the segmented observation and a predefined target pattern is input to a reinforcement learning agent instead. The reinforcement learning agent determines the next action according to the provided input. Here, the determined action is the XY coordinate on the workpiece at which the next virtual laser pulse should be applied.
3) A solid circular shape is rendered at the determined XY coordinate on the virtual workpiece, and the reward earned is calculated and granted to the reinforcement learning agent.

The primary difference between the virtual and physical environments is that the image processing of raw camera images is not required in the virtual environment. This is because the segmented observation of a physical workpiece (i.e., the feature image) can be calculated directly without needing to simulate a camera observation and then applying the image processing procedure.

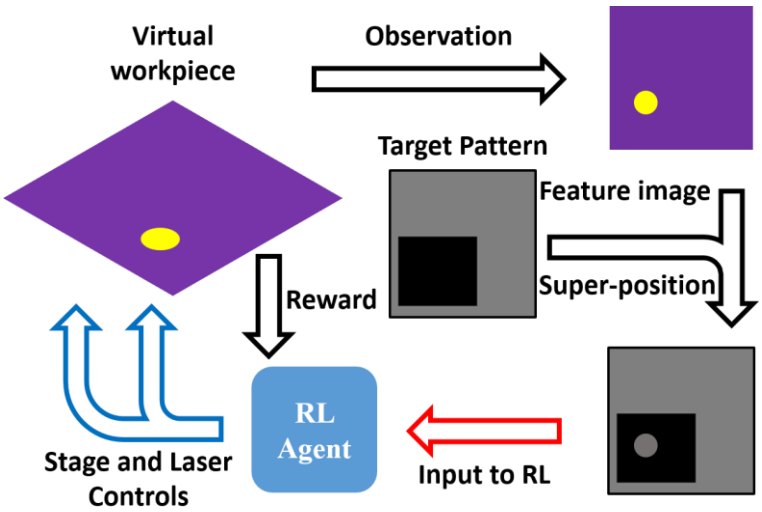

Figure 7-4 Schematic of the virtual environment.

The granted reward at the end of each time step encourages the reinforcement learning agent to learn a set of preferred behaviours. In this study, the set of preferred behaviours is adaptively selecting XY coordinates on a workpiece jointly based on the observation of the workpiece and the predefined target pattern. As discussed in previous chapters (Section 3.4 and 6.4), a suitable reward structure is critical for a reinforcement learning agent to learn a set of preferred behaviours.

As mentioned previously, the image input to the reinforcement learning agent contains only two permitted values c^b and c^t, representing 'background' areas and 'remaining target' areas respectively. Intuitively, in order to incentivise a reinforcement learning agent to laser machine the 'remaining target' areas rather than the 'background' areas, it is sensible to reward a reinforcement learning agent when it machines a portion of the 'remaining target' area, and to penalise it when it machines a portion of the 'background' area. Let a feature image be mathematically described by a matrix $\mathcal{O} = \mathbf{C}^{m\times n}$, where the superscript $m \times n$ specifies the dimensions of this feature image, and where $\mathbf{C} = \{c^b, c^t\}$ is the set that contains the two permitted values c^b and c^t. If we assume that the dimensions of the feature image are such that $m = n$, then an action α determined by the reinforcement learning agent can be denoted as the set of two element matrices $\alpha = \mathbf{N}^2$, where $\mathbf{N} = \{i | i \in \mathbb{Z} \text{ and } 0 < i \leq n\}$ consists of positive integers between 0 and n. (I.e., α is essentially the XY coordinate on the workpiece at which a pulse is incident). Applying an action α to a workpiece causes a circular laser-machined structure centred at α to appear on the workpiece, which then causes a circular area, centred at α, to appear on the feature image of the workpiece. Let all the coordinates within this circular area, centred at α on the feature image, be described by the set $\mathcal{O}^\alpha$, and let the two subsets of this set, $\mathcal{O}_b^\alpha = \{i \big| i \in \mathcal{O}^\alpha \text{ and } i = c^b\}$ and $\mathcal{O}_t^\alpha = \{i | i \in \mathcal{O}^\alpha \text{ and } i = c^t\}$ be the sets that contain the coordinates within $\mathcal{O}^\alpha$ that are of value c^b and c^t respectively. Obviously, these two subsets satisfy $\mathcal{O}_b^\alpha \cup \mathcal{O}_t^\alpha = \mathcal{O}^\alpha$ and $\mathcal{O}_b^\alpha \cap \mathcal{O}_t^\alpha = \emptyset$. The cardinalities of these subsets denoted $\boldsymbol{n}(\mathcal{O}_b^\alpha)$ and $\boldsymbol{n}(\mathcal{O}_t^\alpha)$, measure the size of each set, and thus in this case, they measure the number of coordinates that possess the value of c^b or c^t in $\mathcal{O}^\alpha$. In other words, they represent the pixelised sizes of the 'remaining target' and 'background' areas within the laser-machined structure $\mathcal{O}^\alpha$, on the feature image. In order to incentivise a reinforcement learning agent to machine the 'remaining target' areas rather than the 'background' areas, as previously discussed, it is intuitive to reward the actions that lead to material removal on the 'remaining target' areas and penalise actions that lead to material removal on the 'background' areas. A reward function can thus be constructed as

$$\mathcal{R}(s = s, a = \alpha) \stackrel{\text{def}}{=} n(\mathcal{O}_t^\alpha)_s - \beta \times n(\mathcal{O}_b^\alpha)_s \qquad (7-1)$$

where $\beta \in \mathbb{R}$ is a hyperparameter which is constrained by $\beta \geq 0$. This reward function yields a net reward that increases with $\boldsymbol{n}(\mathcal{O}_t^{\alpha})$ and decreases with $\boldsymbol{n}(\mathcal{O}_b^{\alpha})$, that is, the reward increases with the pixelised size of the machined area on the 'remaining target' and decreases with the pixelised size of the machined area on the 'background' region of the feature image. Note that, in order to be consistent with the previously defined notation of the reward function $r = \mathcal{R}(s, a)$, a subscript s is added to the cardinality operator $\boldsymbol{n}(\cdot)$, which indicates that the corresponding piece of reward r is granted to a reinforcement learning agent at the state s.

The bad exit condition was previously defined as the action that does not contribute to the completion of laser machining the shape of a target pattern. Here, the bad exit condition can be more rigorously described as an action that leads to the piece of reward $r = \beta \times \boldsymbol{n}(\mathcal{O}_b^{\alpha})$. The absence of $\boldsymbol{n}(\mathcal{O}_t^{\alpha})$ here means that no coordinates of value c^t are present within the extent of the laser-machined structure $\mathcal{O}^{\alpha}$ in the feature image. This in turn, means that material removal is performed only from 1) the surrounding material on the workpiece, or 2) areas where laser pulses have previously been applied. Similarly, the good exit condition, which was previously defined as the action that completely machines the target shape on the workpiece, can be described mathematically in the same manner, as the action that leads to the feature image of the workpiece that does not possess any elements with the value of c^t i.e., $\mathcal{O} = \{c^b\}^{n \times n}$ (assuming $m = n$).

This scheme of terminating an episode does not explicitly prevent a reinforcement learning agent from machining outside of the 'remaining target' areas with an incident pulse, provided that this incident pulse also machines some portion of the 'remaining target' area. Therefore, in some cases, for example where a target pattern contains narrow features that are thinner than the diameter of the laser-machined structures, the exact shape of the target pattern cannot be replicated on the workpiece via these laser-machined structures. This makes intuitive sense, as if the features to be machined that are smaller than the laser focus, then it is not possible to machine them without also machining regions outside of the feature. Hence, in the decision-making process of the reinforcement learning agent, when it is machining a thin feature, some trade-offs may need to be made in order for the reinforcement learning agent to obtain the highest possible long-term reward. The reinforcement learning agent may need to deliberately machine marginally outside of thin features in the target pattern in pursuit of a high episodic reward, whilst machining too much extraneous material would tend to lower episodic reward. Due to this, the fidelity of the merged shape of laser-machined structures, with reference to the shape of a predefined target pattern, should be maximised by maximising the episodic reward. From equation 7-1, the undiscounted episodic reward of an episode (see equation 3-5) that is halted by the good exit condition can be written as:

$$R = \mathcal{R}(s_1, a_1) + \mathcal{R}(s_2, a_2) + \cdots + \mathcal{R}(s_n, a_n)$$

$$= n\left(\mathcal{O}_t^{\alpha_1}\right)_{s_1} - \beta \times n\left(\mathcal{O}_b^{\alpha_1}\right)_{s_1} + n\left(\mathcal{O}_t^{\alpha_2}\right)_{s_2} - \beta \times n\left(\mathcal{O}_b^{\alpha_2}\right)_{s_2} + \cdots$$
$$+ n\left(\mathcal{O}_t^{\alpha_n}\right)_{s_n} - \beta \times n\left(\mathcal{O}_b^{\alpha_n}\right)_{s_n}$$
$$= n(\mathcal{O}_t) - \beta \sum_{t=1}^{n} n\left(\mathcal{O}_b^{\alpha_t}\right)_{s_t} \quad (7-2)$$

where $\mathcal{O}_t = \{i | i \in \mathcal{O} \text{ and } i = c^t\}$ is the set of coordinates in the initial target pattern that are of value c^t, and thus $\boldsymbol{n}(\mathcal{O}_t)$ measures the initial pixelised size of the target pattern. Since $\boldsymbol{n}(\mathcal{O}_t)$ only varies with the initial target pattern, the maximum obtainable undiscounted episodic reward is upper bounded by $\boldsymbol{n}(\mathcal{O}_t)$ for a given target pattern. In addition, if any incident pulse is applied at the surrounding material, and/or areas where incident pulses have been previously applied, this makes a negative contribution to overall reward and tends to lower the undiscounted episodic reward. However, the fidelity of the merged shape of laser-machined structures, with reference to the shape of the target pattern, is only irreversibly decreased if the incident pulse is applied at the surrounding material. Therefore, a high undiscounted episodic reward can be regarded as a sufficient but not a necessary condition for high fidelity of the merged shape of laser-machined structures, with reference to the shape of the target pattern. In other words, a high undiscounted episodic reward is a result of machining in accordance with a predefined target shape, with minimal spatial overlap between the applied pulses (and thus few negative reward contributions); whereas, 1) frequently machining the surrounding area (i.e., machining not in accordance with a predefined target pattern), and/or 2) densely overlapped incident pulses give rise to a low undiscounted episodic reward. Here, it is manifestly clear that a high undiscounted episodic reward is not a necessary condition for a high fidelity — if the reinforcement agent decides to machine in accordance with the target pattern through applying densely overlapped incident pulses, it is possible for the reinforcement learning agent to achieve a high fidelity while simultaneously having a low undiscounted episodic reward. Nevertheless, the proposed reward structure is satisfactory as the figure of merit for a reinforcement learning agent to learn the required skills; since maximising undiscounted episodic reward leads to laser-machined structures whose merged shape is conforms to that of the predefined target pattern with a high fidelity.

In theory, the hyperparameter β can be any positive real number: the undiscounted episodic reward is always capped at the pixelised area of a predefined target pattern (i.e., $\boldsymbol{n}(\mathcal{O}_t)$), and a larger β value only increases the variance of the returned pieces of reward. But owing to the stochastic nature of the optimisation process, a very large or a very small β could likely lead to very large or very small pieces of reward, which in turn could give rise to numerical instability (e.g., exploding gradient). Therefore, $\beta = 1$ was adopted in the remainder of this chapter. Additionally, in order to further reduce the variance of the reward function, a normalisation term is added to equation 7-1 to yield the normalised reward function as

$$\mathcal{R}(s = s, a = \alpha) \overset{\text{def}}{=} \frac{n(\mathcal{O}_t^{\alpha})_s - n(\mathcal{O}_b^{\alpha})_s}{n \times n} \qquad (7-3)$$

in which n is the length of the visible workpiece, and $n \times n$ is therefore the total pixelised area of the observable workpiece (assuming that the visible workpiece is of a square shape). Lastly, it is of great importance to point out that, in equation 7-2, all components in the summation are not discounted (see equation 3-6), which is not realistic in the studies of reinforcement learning. However, the theoretical justification of the proposed reward function becomes excessively more sophisticated if the discount factor is included into the discussion, owing to the fact that the upper-bound of a discounted episodic reward is also a function of the step count:

$$\boldsymbol{n}\left(\mathcal{O}_t^{\alpha_1}\right)_{s_1} + \gamma \boldsymbol{n}\left(\mathcal{O}_t^{\alpha_2}\right)_{s_2} + \cdots + \gamma^n \boldsymbol{n}\left(\mathcal{O}_t^{\alpha_n}\right)_{s_n} < \boldsymbol{n}\left(\mathcal{O}_t^{\alpha_1}\right)_{s_1} + \boldsymbol{n}\left(\mathcal{O}_t^{\alpha_2}\right)_{s_2} + \cdots + \boldsymbol{n}\left(\mathcal{O}_t^{\alpha_n}\right)_{s_n}$$
$$= \boldsymbol{n}(\mathcal{O}_t), 0 < \gamma < 1$$

Therefore, a near-unity discount factor (γ=0.997) was adopted in, order to approximate the proposed reward structure. A near-unity discount factor is of common use in the studies of reinforcement learning (as mentioned in the previous chapter at Section 0), and furthermore, partial observability is not a concern in the physical and virtual environments of this chapter (see Section 0).

7.5 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The previously described virtual environment, for a fixed target pattern, can be treated as a black-box optimisation problem, sometimes also known as a derivative-free optimisation problem. The former name gives this type of problem a vivid, colloquial summary — solving a puzzle without looking at the puzzle; the latter name, on the other hand, provides a more rigorous overview of these problems — maximising an objective function $J(x)$ without accessing the derivative information ${\partial J(x)}/{\partial x}$. Note that, rigorously, nuances exist between the formal definitions of the black-box optimisation and the derivation-free optimisation. However, these nuances are beyond the scope of this thesis, and therefore the terms ‘black-box optimisation’ and ‘derivative-free method’ are used interchangeably here (Audet and Hare, 2017). A wide range of algorithms have been developed over the past few decades in order to address these problems, for instance, Simulated Annealing (Bertsimas and Tsitsiklis, 1993) and Nelder-Mead method (Nelder and Mead, 1965). And therefore, it is viable to compare the results for a reinforcement learning agent with the results for a black-box optimisation algorithm, given the same target pattern, in order to quantitively benchmark the capability of the reinforcement learning agent. Amongst the wide spectrum of black-box optimisation algorithms, the well-studied, well-performing Covariance

Matrix Adaptation Evolution Strategy (CMA-ES) algorithm (Hansen, 2016) was chosen for comparison with the reinforcement learning agent.

The Covariance Matrix Adaptation Evolution Strategy belongs to a greater class of direct search algorithm known as the Evolution Strategy (ES). The Evolution Strategy algorithm is biologically inspired by the Darwinian paradigm of evolution and is composed of four phases in a never-ending loop, namely Parents, Reproduction, Offspring and Selection (Beyer, 2001). The Parents possess some inherent, genetic-coded attributes that allow them to survive long enough in an environment in order to pass these attributes on to their Offspring via Reproduction. In the process of Reproducing, the genetic information from the Parents is subject to replication, recombination and mutation, which in turn may causes their Offspring to inherit some of their genetic-coded attributes, or to accidentally develop new attributes. This herd of Offspring, with their distinctively varying attributes, are then trialled in the very same environment; the ones with the attributes that are better-suited for the environment have a higher chance to survive longer and eventually become the next generation of Parents, who are able to pass their genetic information to their Offspring. In contrast, the Offspring who possess inferior attributes amidst the environment are less likely to live long enough to participate in Reproduction, and thus their lower fitness attributes are less likely to be passed to the next generation of the Offspring.

In a similar context, let $y = J(x)$ be the objective function to be optimised, and the inputs x_i of shape n are all real numbers (i.e., $x_i \in \mathbb{R}^n$). The inputs x_i are parameterised by an n-dimension isotropic Gaussian distribution $x_i = \{x_m | x_m = \mathcal{N}(\mu, \sigma I), m = 0,1,2,\cdots,n-1\}$ (i.e., Parents). In order to generate a herd of offspring of size Λ, a candidate set of size Λ which is composed of x_i can be generated by sampling from the Gaussian distribution Λ times $D^t = \{x_i^t | x_i^t = \mu^t + \sigma^t y_i^t \text{ where } y_i^t = \mathcal{N}(0, I), i = 0,1,2,\cdots,\Lambda-1\}$ (i.e., Reproducing and Offspring), where the superscript t denotes, here, the number of iteration. The candidate set then can be trialled by the objective function $y_i = J(x_i)$, and the first Ω candidate solutions (obviously $\Omega < \Lambda$) with the highest respective values of y constitute the 'winning' set $W_t = \{x_i^t | x_i^t \in D'^t, i = 0,1,2,\cdots,\Omega-1\}$ (i.e., Selection), where D'^t is the re-indexed sequence of D^t based on $y_i = J(x_i)$. The parameters μ^t and σ^t can therefore be updated from the 'winning' set $W_t - \mu^{t+1} = \frac{1}{\Lambda}\sum_{i=0}^{\Omega} x_i^t$ and $\sigma^{t+1^2} = \frac{1}{\Lambda}\sum_{i=0}^{\Omega}(x_i^t - \mu^t)^2$. This process is repeated until the average value $\hat{y}$ of the candidate set D^t exceeds a predefined threshold. The CMA-ES introduces a Gaussian distribution with a covariance matrix C that replaces the isotropic Gaussian distribution to parameterise the inputs (i.e., $x_i = \{x_m | x_m = \mathcal{N}(\mu, \sigma C), m = 0,1,2,\cdots,n-1\}$). This addition improves the exploration of ES (as exploration of each input is separately and independently controlled by a covariance matrix), and therefore enhances its applicability to more ill-conditioned

functions (details concerning adaptive step size control and theoretical justification of the CMA-ES algorithm can be found in (Hansen, 2016)).

CMA-ES is commonly utilised to perform parameter optimisation in a continuous domain for non-linear, convex functions, that is, to search for sets of parameters $A = \{a_0, a_1, a_2, \cdots, a_n\}$ that reach a local maximum $y_{\max}$ or minimum $y_{\min}$ in a non-linear and convex function $y = f(a_0, a_1, a_2, \cdots, a_n)$. Whilst CMA-ES is not deliberately designed for problems where actions are taken consecutively, like the previously described virtual and physical environment, a set of actions $A = \{a_0, a_1, a_2, \cdots, a_n\}$ can be generated simultaneously in advance prior to the first step in the environment, and thereafter each individual action a_i from A can be inputted to the environment in a sequential manner, at each step, until one of the exit conditions is met. (The length of A needs to be large enough for the CMA-ES agent to encounter an exit condition, and thus the length of A is uniformly set to 100 for all the CMA-ES optimisations for the remainder of the chapter). The discounted episodic reward is then returned to the CMA-ES agent for optimising the set of actions. Notably, the episodic reward function (i.e., function 7-2) is regarded as an ill-conditioned concave function to be maximised here (it has been previously demonstrated that the undiscounted episodic reward function is upper bound by the pixelised area of the target pattern). Comparatively, the similarities between a reinforcement learning agent and a CMA-ES agent are that:

1) They both solely rely on the granted reward from the virtual environment to perform optimisation.
2) The virtual environment processes the selected XY coordinates from both decision-making agents in an identical manner.

On the other hand, the differences between these two decision-making agents are that:

1) The reinforcement learning determines one action (i.e., one XY coordinate) per step, whereas the CMA-ES agent determines all actions simultaneously, given a predefined target pattern, prior to the start of an episode in the virtual environment.
2) The determined actions (i.e., XY coordinates) from the reinforcement learning agent are all based on observations of the workpiece at corresponding time steps, whereas, for the CMA-ES, actions (i.e., XY coordinates) are generated without any reference to observations of the workpiece.
3) Provided that it is trained accordingly, it is possible for the reinforcement learning agent to be applied to multiple different target patterns, whereas, every newly-encountered target pattern requires a separate instance of CMA-ES optimisation to search for a set of XY coordinates.

Whilst the two decision-making agents differ in terms of their optimisation mechanics, the sets of XY coordinates that these two decision-making agents generate can be compared directly, owing to the fact that both decision-making agents utilise the granted reward as the sole source of information for their respective optimisations, and that the XY coordinates which these two decision-making agents generate are handled indistinguishably by the virtual environment. This enables a comparative, quantitative assessment of the performance of a reinforcement learning agent.

For clarity, the CMA-ES algorithm is a well-optimised, very-capable optimisation algorithm, and thus the set of XY coordinates that it finds for a target pattern should be very close to, if not actually, the optimal solution for a given target pattern. And thus, it is anticipated that the CMA-ES agent could marginally outperform the reinforcement learning agent, which offers greater generality. However, it needs to be clearly stated that the CMA-ES is a black-box optimisation algorithm, and consequently one instance of CMA-ES optimisation solely searches for a single set of XY coordinates, applicable to just one particular target pattern. For the reinforcement learning agent, on the other hand, the underlying problem is not treated as a black-box optimisation, in the sense that the reinforcement learning agent has access to derivative information. Owing to this scheme of optimisation (via derivative information), the reinforcement learning agent is capable of selecting XY coordinates for more than a single target pattern. The critical difference here is that the two decision-making agents are tasked to solve problems of different complexities. Therefore, the possibly inferior results from the reinforcement learning agent, compared to results from the CMA-ES agent for a single fixed target pattern, can be explained by the greater complexity of the problem tackled by the reinforcement learning agent.

The CMA-ES algorithm is therefore introduced to this study solely as a means to estimate the performance upper bound. As discussed previously, target patterns may contain narrow features which are thinner in dimensions than the diameter of the laser-machined structures. This makes it extremely challenging to estimate the theoretical maximum episodic reward achievable for these target patterns, owing to the fact that machining these features would inevitably also result in machining some portion of the surrounding material. The CMA-ES algorithm therefore provides a relatively applicable and convenient means to estimate the maximum episodic reward that is achievable by a decision-making agent (i.e., a performance upper bound). The aim of demonstrating the reinforcement learning agent in the virtual and the physical environment is hence not to surpass the CMA-ES agent in its ability to select appropriate XY coordinates. Instead, the motivation of this study is to demonstrate the following abilities of the reinforcement learning agent:

1) The ability to determine actions (i.e., XY coordinates) for any arbitrary target pattern, without any the need for additional optimisation.
2) The ability to detect any inadvertent errors in the positioning of laser machined structures, and to adjust the subsequent actions to compensate for this.

The results for the CMA-ES agent, therefore, provide supplementary information, helping us to assess the discrepancy between the results for a reinforcement learning agent and the optimum solution.

7.6 Application to a Fixed Target Pattern

The fundamental objective throughout the studies in this chapter is to find a generalised reinforcement learning agent that is capable of finding a set of XY coordinates $A = \{a_0, a_1, a_2, \cdots, a_n\}$ that maximises the discounted episodic reward for the environment, given any arbitrary target pattern. A primary challenge here is that the generality of a reinforcement learning model is difficult to evaluate, owing to the fact that it is impractical to benchmark a reinforcement learning agent with all possible target patterns that can be fit onto the observable area of a workpiece. This challenge is further exacerbated by the fact that the adopted reinforcement learning algorithm, PPO, employs neural networks as function approximators, which are, by design, highly non-linear. This makes it infeasible to evaluate the generality of the reinforcement learning agent by analytically examining its actuating components (the neural networks). To tackle this problem, it is first demonstrated that the reinforcement learning agent is applicable to a simplified virtual environment in which the requirement for a single reinforcement learning agent to be able to process any arbitrary target pattern is relaxed. In other words, the predefined target pattern is kept constant in this simplified virtual environment, and the reinforcement learning agent only needs to learn a dedicated policy (i.e., a set of preferred behaviours) for this fixed target pattern. This simplification (i.e., to keep the predefined target pattern unchanged) degrades the previously described adaptive XY coordinate selection problem into a black-box optimisation problem. That is, in this simplified environment, accessing the derivative information (and consequently observing the workpiece in real time) is no longer a necessity in order to search for a set of XY coordinates that maximises the discounted episodic reward.

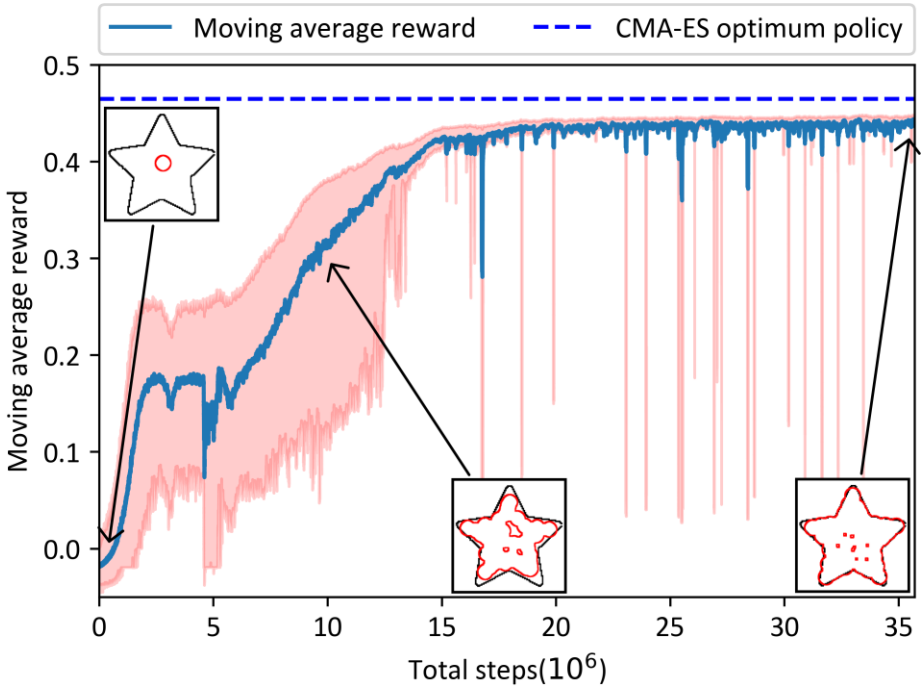

Figure 7-5 Training curve of the virtual environment with fixed target pattern. The target pattern chosen here is a five-pointed star. The three insets show the performance of the reinforcement learning at the respective training step. The blue dashed line shows the theoretical performance upper bound, which is obtained from an instance of CMA-ES optimisation, given the same target pattern.

Figure 7-5 shows the training curve, defined as the moving average of undiscounted episodic reward (with a window size of 1000 episodes, see equation 3-5) versus the total number of steps, for the virtual environment where the target pattern is kept constant as a five-pointed star. At the start of training, between 0 to 15×10^6 steps, the progress in learning a policy for selecting appropriate XY coordinates was very rapid, while the large variance in undiscounted episodic reward also indicates that the reinforcement learning agent endeavoured to find an optimum policy through investigating a large variety of different policies. On the other hand, after 15×10^6 steps, in the later stage of training, the undiscounted episodic reward plateaued at a level that is marginally lower than that of the optimal solution (estimated by an instance of CMA-ES optimisation using the same five-pointed star as the target pattern). In addition, the variance of undiscounted episodic reward was significantly reduced in the later stages of training, signalling that the reinforcement learning agent has found a policy which it considers to be close to optimal. This policy was further refined after 15×10^6 step, and the moving average undiscounted episodic reward reached its apex at approximately 36×10^6 steps.

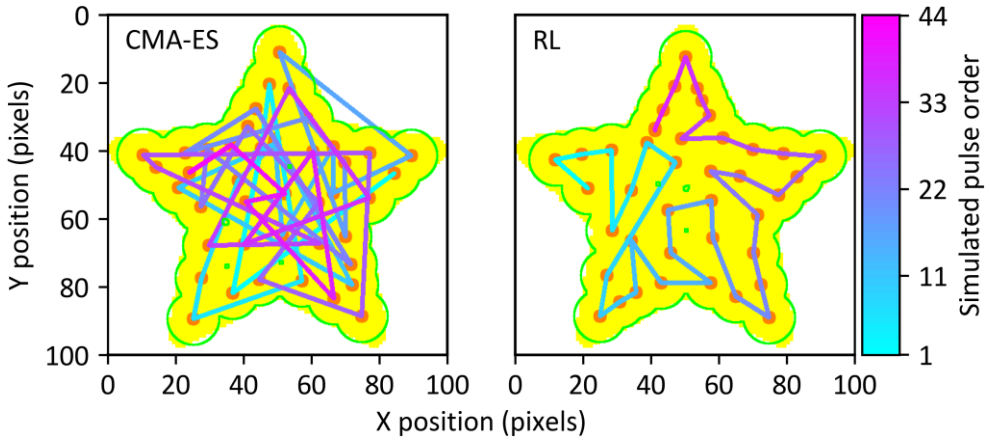

Figure 7-6 A comparison between the result obtained from the CMA-ES optimisation and the result obtained by taking the mean of the Gaussian distributed policy of the reinforcement learning agent, given the same five-pointed star target pattern.

This training curve highlights the discrepancy between the result for the optimal policy found by the CMA-ES agent and the result for the reinforcement learning agent, indicating that the reinforcement learning agent has not matched the result of to the CMA-ES agent. On the face of it, this is a fair comparison, in the sense that the policies of both the reinforcement learning agent and the CMA-ES agent are parameterised by Gaussian distributions, and they both utilise pieces of reward generated by the same scheme of reward-granting, from the same environment, as their source of information. However, the optimal policy found by the CMA-ES agent is by definition the candidate solution with the highest undiscounted episodic reward from the candidate set. This cherry-picking 'highest of the highest' approach for determining the optimum policy for the CMA-ES algorithm is purposely chosen in order to estimate the theoretical upper bound for the undiscounted episodic reward. This comparison is thus disadvantageous to the reinforcement learning agent, owing to the fact that the moving average of undiscounted episodic reward does not allow cherry picking of data. A more equitable comparison would require additional sampling for the reinforcement learning agent, allowing cherry picking of the best results, however, this would also increase the time complexity of the reinforcement learning algorithm. Therefore, instead of pursuing absolute fairness in comparing the results from the CMA-ES agent and the reinforcement learning agent, a less equitable but more practicable comparison is made in Figure 7-6, where the trajectory of the selected XY coordinates from the CMA-ES optimum policy is shown on the left-hand side, and the trajectory of the mean μ of the Gaussian distributed policy from the reinforcement learning agent is shown on the right-hand side. The CMA-ES agent scores 0.465 undiscounted episodic reward in total, using 45 laser pulses to successfully machine 97.55 % of the target pattern but inadvertently machines 1.93 % of the target pattern area from the surrounding material. In comparison, the reinforcement learning agent scores 0.462 undiscounted episodic reward in total, using just 40 laser pulses to successfully machine 96.85 % of the target pattern

whilst inadvertently machining 1.72 % of the target pattern area from the surrounding material. Although, in this comparison, the CMA-ES agent still has the advantage of cherry picking its best result, the discrepancy between the results for these two agents are extremely marginally. It can be observed that, in terms of the percentage of the target pattern machined and the undiscounted episodic reward, the CMA-ES agent performs slightly better than the reinforcement learning agent; however, it should also be noticed that the reinforcement learning agent uses four fewer pulses than the CMA-ES agent. Thus, it is evident that, at least for the five-pointed star target pattern, the policy found by the reinforcement learning agent is comparable with that of the CMA-ES agent.

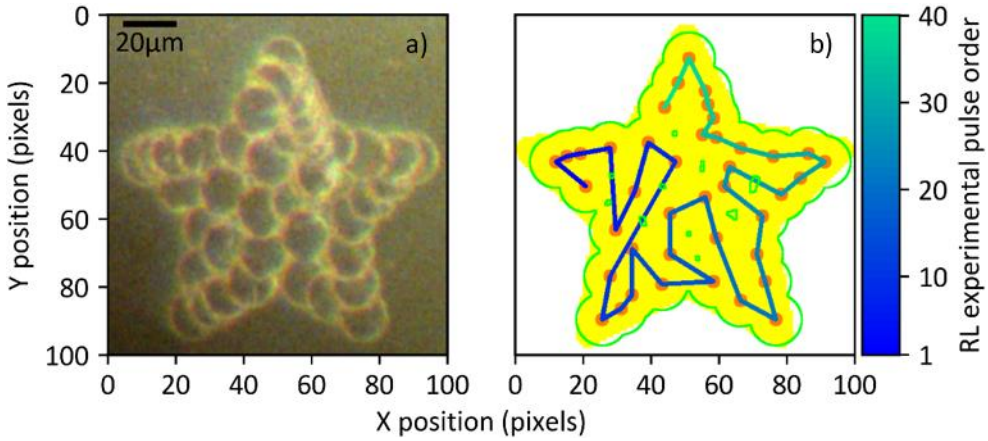

Figure 7-7 a) The camera observation of the experimental result. b) The associated visualisation of the rollout of the experiment.

The reinforcement learning agent was then applied to the physical environment — as discussed previously, the reinforcement learning agent is operated in a manner similar to that employed in the virtual environment; the primary difference is that in the physical environment the input to the reinforcement learning agent involves image processing of camera observations of a workpiece (i.e., 1 mm thick fused silica microscope slide). At each discrete time step, the output of the reinforcement learning agent was used to control both the pulse firing and the movements of the XY translation stage that held the workpiece. The camera observation of the experimental result is shown on the Figure 7-7 a), whilst the trajectory of selected XY coordinates obtained from image processing is shown on the Figure 7-7 b). In this experiment in the physical environment, the mean of the Gaussian distributed policy μ from the reinforcement learning agent was used, effectively rendering the output of the reinforcement learning agent deterministic. It is therefore possible to directly compare the trajectory of selected XY coordinates from the physical environment with those selected in the virtual environment. For example, we can compare the simulation trajectory of the virtual environment in Figure 7-6 (right) and the experimental trajectory of the physical environment in Figure 7-7 a), and we can observe that these two trajectories are not identical (although they are indeed rather similar). In fact, not only does the distribution of selected XY coordinates of the experimental trajectory differ from that of the virtual environment trajectory,

but an additional pulse was also applied to the workpiece in the physical environment (i.e., Figure 7-7). The origins of these nuanced differences in trajectory are twofold:

1) In the physical environment, varying illumination conditions and visual changes in the vicinity of the laser-machined structures (e.g., laser-induced shock waves that blow small fragments off the workpiece) can give rise to inaccurate segmentation of the laser-machined structure, which in turn causes the reinforcement learning agent to deviate from its determined optimum trajectory.
2) Backlash, drift and vibrations that stem from the XY translation stages can lead to inaccurate positioning of the workpiece, which can cause the laser pulses to be incident at a slightly different position to that desired by the reinforcement learning agent.

Whilst these software and hardware limitations, that are only present in the physical environment, disturbed the rollout of the optimum trajectory determined by the reinforcement learning agent, the reinforcement learning agent was still able to detect and consequently compensate for any inadvertent pulse positioning errors, leading eventually to the complete machining of the target pattern onto the workpiece. This 'self-correction ability' is further discussed in Section 7.10, and is a key advantage of using a reinforcement learning approach. In this experiment in the physical environment, the mean μ of the Gaussian distributed policy from the reinforcement learning agent was used (hence preventing random exploratory actions).

As per the discussion, for the remainder of this chapter, if not stated otherwise, all trajectories related to the reinforcement learning agent should be assumed to be obtained through taking the mean μ of the Gaussian distributed policy, and all trajectories related to the CMA-ES agent should be assumed to be the best performing candidate solution (i.e., highest undiscounted episodic reward) found in the candidate set with the highest average undiscounted episodic reward.

7.7 Application to a Rotating Pattern

The necessity to observe the state of an environment in order to determine an appropriate action commonly arises when the dynamics of the environment is subject to randomness. That is, a randomly presented initial state and state transitions that are of a random nature typically give rise to the need to observe the state of an environment. Theoretically speaking, the randomness is of less common occurrence in the physics of light-matter interactions. However, it is still possible for the hardware and software that are responsible for the controlling and monitoring of a machining process to malfunction, which could in turn introduce a degree of randomness to laser materials processing. The ability to rapidly generate toolpaths in accordance with desired target patterns, for

laser materials processing, even in the presence of some randomness, is attractive to researchers and manufacturers, in the sense that this ability enables rapid prototyping with robustness against some forms of inadvertent error. The ability to generate a toolpath as per a desired target pattern can be regarded as the ability to select appropriate XY coordinates for a desired target pattern, in the context of this study. This differs from the simplified virtual environment described on the previous section, where the target pattern is invariable. In this section, the reinforcement learning agent is tasked to perform XY coordinate determination in a more difficult virtual environment, in which the target pattern can be rotated to any angle. This additional complexity introduces a degree of generality, though this generality is only amongst a small number of target patterns (i.e., a target pattern rotated arbitrarily to all angles). By employing the same five-pointed star used in the previous chapter as the target pattern for this environment, the results for the reinforcement learning agent can be compared with those of the previous chapter; this comparison can serve as a benchmark to illustrate the impact that introducing generality has on the performance of the reinforcement learning agent.

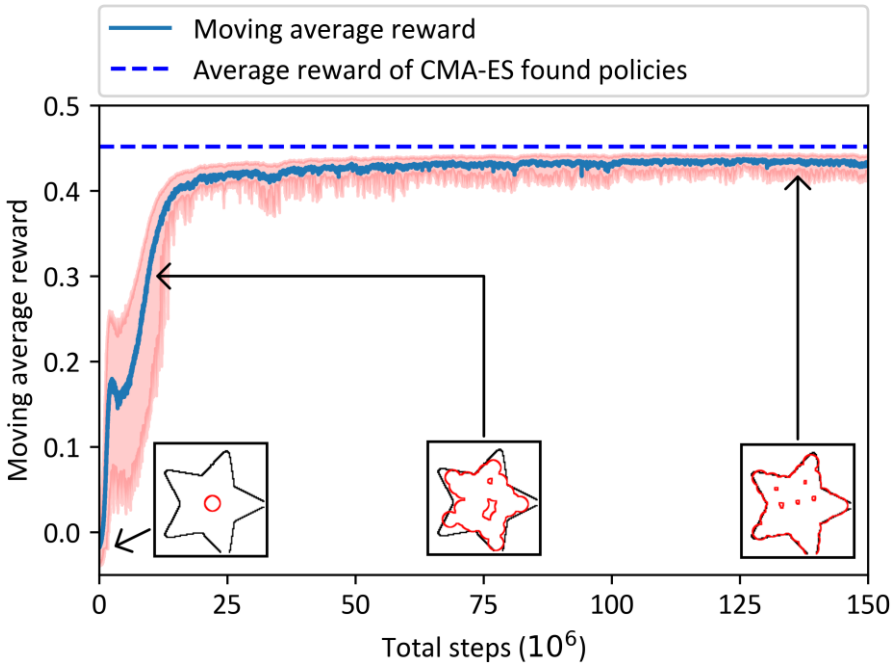

Figure 7-8 Training curve from the virtual environment with the rotating target pattern. The rotating target pattern here is a five-pointed star, which is randomly rotated about its image centre at the start of each training episode. The three insets show the performance of the reinforcement learning at the respective training step (with the same star orientation selected for clarity). The blue dashed line shows the theoretical performance upper bound, which is obtained by averaging the results of 360 instances of CMA-ES optimisation, given 360 target patterns, rotated at each integer angle from 0 to 359 degrees.

In this virtual environment, the five-pointed star target pattern was rotated about its image centre by a random angle at the start of each episode, this rotation angle was drawn from a uniform distribution between 0 and 360 degrees (i.e., $\mathcal{U}_{[0,360)}$, floating point). The training curve is present in Figure 7-8. This figure shows that the progress of the reinforcement learning was rapid before 25×10^6 steps, while a large variance was also observed. And after 25×10^6 steps, the pace of learning of the reinforcement learning greatly reduced, and eventually the peak performance (i.e., highest average undiscounted episodic reward) of the reinforcement learning agent was reached at approximately 136×10^6 steps. For a comparison, 360 instances of CMA-ES optimisation were performed for the same five-pointed star rotated from 0 to 359 integer degrees with an increment of one degree. The average of these results is shown as a blue line in Figure 7-8. In this figure, it is shown that the result for the reinforcement learning was slightly lower than the average result for the CMA-ES found policies, however, the previous discussion in regard to the fairness of the comparison between results for the reinforcement learning agent and the CMA-ES agent can still be applied here. The results in Figure 7-8 can be compared with Figure 7-5, and show that, understandably, the reinforcement learning agent required a larger number of steps to reach a plateaux when the target shape was allowed to rotate to any arbitrary angle.

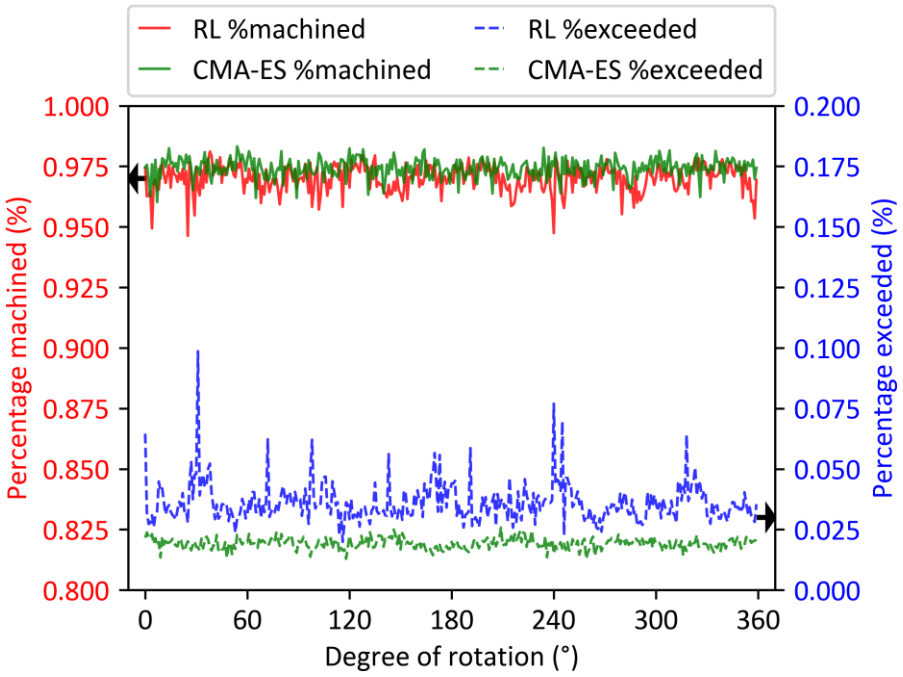

Figure 7-9 A degree-by-degree comparison between the results for the CMA agent and the results for the reinforcement learning agent.

A more in-depth and quantitative assessment of the results for these two agents is shown in Figure 7-9, where a degree-by-degree comparison between the results for the reinforcement learning agent (deterministic output i.e., the mean μ of the Gaussian distributed policy) and the results for the CMA-ES agent is made, in the case for the rotating shape. In this comparison, the

two agents are tasked to perform XY coordination deduction for the same five-pointed star target pattern rotated to every integer angle from 0 to 359 degrees (i.e., 360 target patterns in total), and the results are plotted against the angle of rotation as shown in the figure. The reinforcement learning agent, on average, scores 0.442 ± 0.005 reward and successfully machines on average 97.01 ± 0.54 % of the target pattern, but inadvertently machines 3.60 ± 0.80 % of the target pattern area from the surrounding material. Comparatively, the CMA-ES agent, on average, scores 0.452 ± 0.002 reward and successfully machines on average 97.44 ± 0.39 % of the target pattern, but inadvertently machines 1.92 ± 0.22 % of the target pattern area from the surrounding material. The average rewards of the reinforcement learning agent and the CMA-ES agent are both lower than their respective rewards reported for the fixed five-point star target pattern, presented in the previous section. Part of this reduction in reward is directly related to the rotations that are applied to the five-pointed star target pattern — rotating the five-pointed star at some angles causes the tips of some arms of the star to move outside of the observable workpiece, which in turns reduces the pixelised area of the five-pointed star target pattern (i.e., $\boldsymbol{n}(\mathcal{O}_t)$). Owing to the fact that the undiscounted episodic reward is upper-bounded by $\boldsymbol{n}(\mathcal{O}_t)$, reducing $\boldsymbol{n}(\mathcal{O}_t)$ may result in a reduction in the undiscounted episodic reward for both the reinforcement learning agent and the CMA-ES agent.

The results reported in this section are comparable to the results reported in the previous section (Section 7.6). Statistically, the discrepancy in average undiscounted episodic reward between the reinforcement learning agent and the CMA-ES agent for the rotating target pattern is ~0.01, which is marginally higher than the discrepancy for the fixed target pattern which was just ~0.003. This is undoubtedly an indication that the increase of arbitrary variation in the target pattern reduces the performance of the reinforcement learning agent. This can be better illustrated visually in Figure 7-9, where an obvious gap (~1.7 %) can be found between the blue dashed line and green dashed line which correspond to the percentages of inadvertent machining of areas outside of the target pattern, for the reinforcement learning agent and the CMA-ES agent respectively. This slight but noticeable discrepancy in performances between the two agents appears to arise from the inclusion of arbitrariness (i.e., sample rotation). However, the results reported here still demonstrate that a single reinforcement learning agent is able to perform appropriate XY coordinate deduction in accordance not only to a single fixed target pattern, but also for target patterns rotated at any angle. From a practical standpoint, this demonstrated ability may not be of particular engineering interest, in the sense that a similar functionality can be achieved via conventional means (e.g., rotating the workpiece or, in the case when rotary stage is inaccessible, performing affine transformation to a known optimum trajectory of XY coordinates).

However, the aim here is to demonstrate the multi-purpose capability (generality) of the reinforcement learning agent.

7.8 Application to an Arbitrary Pattern

The arbitrariness amongst a small set of target patterns can give rise to a situation where a decision-making agent can memorise the encountered target patterns and optimises dedicated policies for these memorised target patterns (as shown in Section 7.6). This arises a concern that this decision-making agent cannot be applied to target patterns that are not included in the small set of possible target patterns, owing to the fact that the learned dedicated policy lacks the generality necessary for deducing appropriate XY coordinates in accordance with a previously unseen target pattern. In this regard, if the size of the set of possible target patterns is large enough that it is impossible for a decision-making agent to repeatedly encounter every target pattern during training, the decision-making agent is therefore forced to learn a general principal to perform toolpath generation in accordance with a desired target pattern, instead of memorising all encountered target patterns and optimising the associated dedicated policies for these commonly encountered target patterns. Here, in this section, a virtual environment is devised, where the target pattern is randomly generated. The random shape generator can provide a near-infinite number of distinct target patterns. The shape generation process is achieved through the following steps:

1) Seven pixels are selected uniformly at random (i.e., $\mathcal{U}_{[0,100)}$, integer numbers) within the observable area of a workpiece which is pixelised to a 100x100 pixel image.
2) These selected pixels are then consecutively connected via parametric curves. In this study, the Bézier curve is used.

Nine examples of target patterns are presented in Figure 7-10, where in each subfigure, the selected pixels, also known as the anchor points, are highlighted in green, and a red hollow circle is plotted in its image centre to illustrate the approximate size of a single laser-machined structure for comparison. To estimate the number of possible distinct target patterns, the total number of ways in which the seven anchor points can be selected without considering the order of this selection needs to be evaluated first. This is a classic combination problem, where seven points scattered on a 100x100 pixel image can be interpreted as selecting (not fetching) seven items in a set that contains 10,000 items. Thus, the total number of this combinations is given by $C_k^n = {}^{n!}/_{k!\,(n-k)!}$, where $n = 10{,}000$ and $k = 7$, and it gives $C_7^{10{,}000} \sim 1.98 \times 10^{24}$. However, some combinations of anchor points lead to the same target pattern once digitized into the image array. In addition, target patterns that are of small sizes are likely to possess features whose dimensions are too narrow to machine, considering the size of a laser-machined structure. Such target patterns

would effectively add noise to the reinforcement learning process, to avoid this they are therefore discarded by the random shape generator. The total number of possible distinct target patterns can be therefore estimated via the Monte Carlo method:

1) A large set of combinations of distinct anchor points can be generated.
2) The target patterns can be created based on these anchor points.
3) The number of target patterns that are excluded due to either being duplications of already generated target patterns or having pixelised area that is smaller than 3,000 pixels can be evaluated.

The ratio between the number of distinct anchor point combinations and the number of excluded combinations can thus be used to estimate the total number of possible distinct target patterns. Through sampling a total number of 1×10^7 distinct combinations of anchor points, this ratio is found to be approximately 0.55, which in turns suggests that the total number of possible distinct target patterns is approximately $0.55C_7^{10,000} \sim 1.09 \times 10^{24}$.

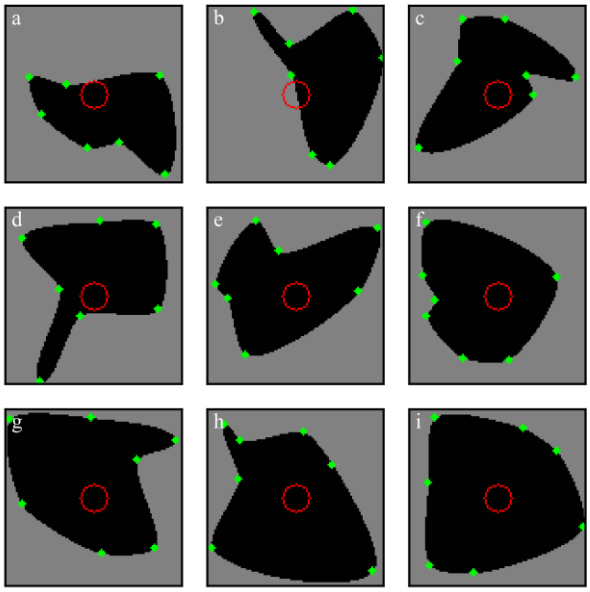

Figure 7-10 Examples of randomly generated target patterns. The scale of a laser-machined structure is shown as a red hollow circular in each subfigure. Some of these target patterns possess thinner features whose dimensions are smaller than the diameter of a laser-machined structure, most notably in b), d) and h).

The training curve for a virtual environment using the random shape generator is presented in Figure 7-11. Similar to the other two instances of training, rapid progress in learning could be seen at the start of the training process between 0 and 50×10^6 steps, whilst the performance slowly and gradually improved thereafter; the apex of performance (i.e., highest average undiscounted episodic reward) was reached at approximately 298×10^6 steps. Unlike the other two instances of training, the variance of the undiscounted episodic reward at the later stage of the

training did not convergence. This is partially because the pixelised area of the target patterns (i.e., $\boldsymbol{n}(\mathcal{O}_t)$) varies considerably between training episodes in this virtual environment, which in turns causes the maximum obtainable undiscounted episodic reward to vary considerably (the maximum obtainable undiscounted episodic reward is upper bounded by $\boldsymbol{n}(\mathcal{O}_t)$). Whilst the CMA-ES algorithm was used in the previous sections as a means to evaluate performance upper bounds of the maximum obtainable undiscounted episodic rewards (see Figure 7-5 and Figure 7-8), it is impractical to do that in this case. In the studies of the fixed target pattern and the rotating target pattern, only 1 and 360 instances of CMA-ES optimisation were needed for estimating these associated upper bounds (i.e., the horizontal dashed lines shown in Figure 7-5 and Figure 7-8). However, in this virtual environment with randomly generated shapes, 1×10^7 target patterns were generated during the training. On average, using CMA-ES, it takes 15 minutes to optimise a single target pattern on a high-end desktop configured with a 9th generation Intel i9 processor. Optimising for all 1×10^7 target patterns in order to evaluate the upper bound with the CMA-ES algorithm would therefore take approximately 290 years. A different approach is therefore adopted in this study to estimate the upper bound of the maximum obtainable undiscounted episodic reward, namely by computing the average of the pixelised area of the generated target patterns (i.e., $\widehat{\boldsymbol{n}(\mathcal{O}_t)}$). Recalling equation 7-2, the undiscounted episodic reward $R \leq \boldsymbol{n}(\mathcal{O}_t)$, given that $\sum_{t=0}^{n} \boldsymbol{n}\left(\mathcal{O}_b^{\alpha_t}\right)_{s_t \to s_{t+1}} \geq 0$. Comparing with the CMA-ES approach for estimating the upper bound, estimating the upper bound through averaging the pixelised area of target patterns (i.e., $\widehat{\boldsymbol{n}(\mathcal{O}_t)}$) results in a higher upper bound, owing to the fact that this estimation does not take into account the unavoidable negative rewards that would be accrued due to machining background regions in the vicinity of thin features. In Figure 7-11, the upper bound estimated via $\boldsymbol{n}(\mathcal{O}_t)$ is shown as a horizontal blue dashed line. In addition, the standard deviation of the pixelised sizes of the generated target patterns is also evaluated and is shown as two horizontal green dashed lines. Notably, comparing with the previously used five-pointed star, the target patterns that are generated by the random shape generator could possess thinner features, which in turn could cause a large discrepancy between this evaluated upper bound and the ground-truth maximum obtainable undiscounted episodic reward. However, since the previous result from Section 7.7 suggests that increasing the generality of the problem faced compromises the performance of the reinforcement learning agent, it should not be surprising here that the much larger set of arbitrary target patterns further reduces the performance of the reinforcement learning agent. Nevertheless, the fact that the training curve is asymptotic to the estimated upper bound indicates that the reinforcement learning agent has learnt, to some extent, the ability to select appropriate XY coordinates in accordance with a desired and arbitrary target pattern.

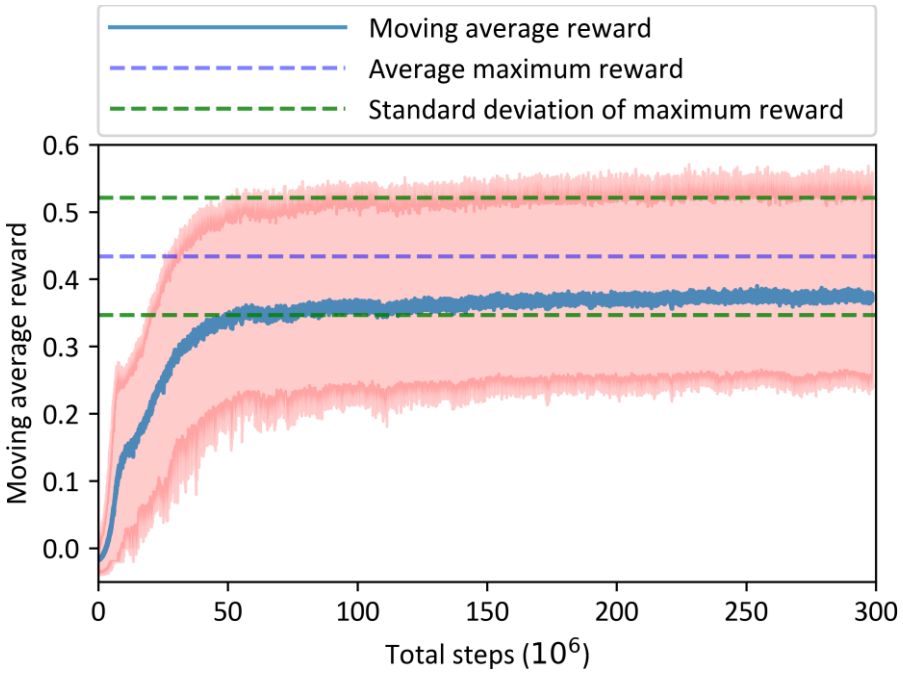

Figure 7-11 Training curve of the virtual environment with arbitrary target patterns. The arbitrary target patterns are randomly generated at the start of each training episode. The blue dashed line shows the estimated maximum undiscounted episodic reward (based on pixelised target area), however this estimated maximum reward may not be obtainable. The two dashed lines indicate the interval of the standard deviation of the estimated maximum undiscounted episodic reward.

A more in-depth assessment of the reinforcement learning agent, in terms of percentage machined and percentage exceeded, is presented in the form of a box-and-whisker diagram in Figure 7-12. A box-and-whisker diagram shows the 0^{th}, 25^{th}, 50^{th} (the median), 75^{th} and the 100^{th} percentile as well as the mean of a distribution of data. In order to investigate the ability of the reinforcement learning agent to perform toolpath generation for randomly generated target patterns, a total number of 140,000 target patterns were generated via the random shape generator. These randomly generated target patterns were sorted based on their pixelised area, and then divided into seven categories, namely 3000~3500, 3500~4000, 4000~4500, 4500~5000, 5000~5500, 5500~6000 and 6000~10000 pixels. Given that the array used to describe each of the shapes was 100 by 100, the maximum total area was 10000. Each of these categories contained a total number of 20,000 target patterns. The reinforcement learning agent (already trained in the arbitrary shape environment) was then used to perform toolpath generation for each of these shapes, statistical analysis of these results is shown in Figure 7-12. It can be seen from the figure that the average percentage of successfully machined target pattern area increases with the size of the target pattern, while the average percentage of the inadvertently machined surrounding area of the target pattern decreases with the size of the target pattern. In addition, the distribution of percentages for both the successful and the inadvertent machining become more symmetric as the

size of the target pattern grows. It is therefore evident that the performance of the reinforcement learning agent increases with the size of the target pattern. This is perhaps due to the fact that it is less likely for a target pattern with a large size of area to possess thin features whose dimensions are smaller than the diameter of a laser-machined structure (~18 µm). Machining a feature that is thinner in diameter than that of a laser-machined structure often results in an insignificant positive reward, and sometimes even a negative reward, this discourages the reinforcement learning agent from machining thin features, which in turn results in a lower percentage of target pattern area being machined for small shapes. And in the cases where such thin features are machined, it is likely that considerable portions of the surrounding material will also be machined, resulting in a high percentage exceeded as shown on the figure.

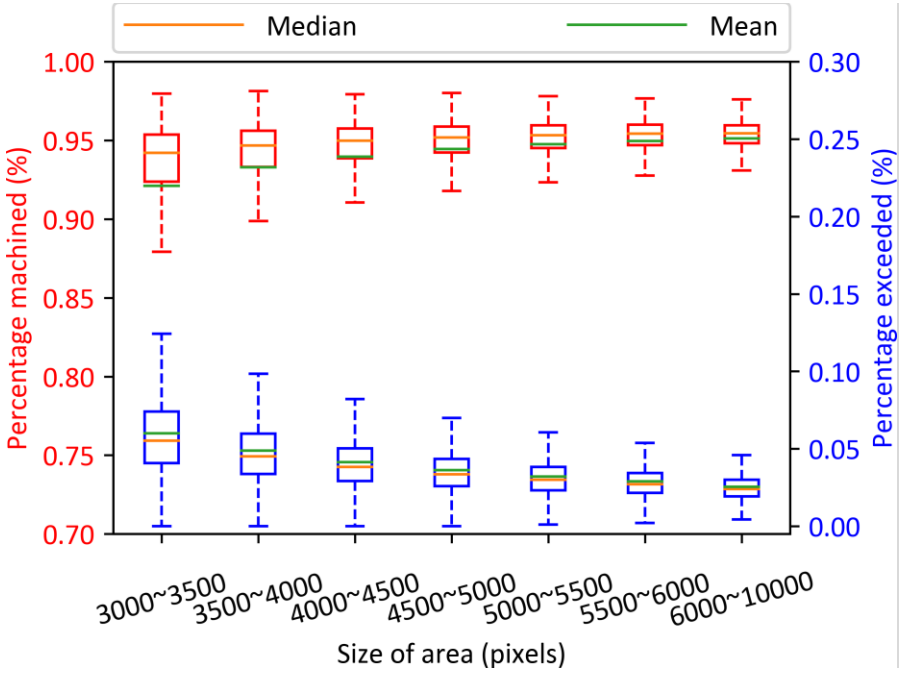

Figure 7-12 A box-and-whisker diagram that shows the distribution of the testing results for the reinforcement learning agent in the virtual environment where the target pattern is randomly generated. The percentage of the target pattern machined is shown as the 'Percentage machined' on left Y axis and the percentage of the surrounding material machined is shown as 'Percentage exceeded' on the right Y axis. The results are categorised by area of the target pattern.

The reinforcement learning agent was then used in the physical environment for two real-world laser machining experiments, in which the two target patterns were randomly generated at the beginning of the associated experiment. The camera observations of these two experiments are shown in Figure 7-13 a) and e), and their respective image processing results are shown in Figure 7-13 b) and f). In the previous discussion, the total number of possible distinct target patterns is estimated to be $0.55C_7^{10,000} \sim 1.09 \times 10^{24}$. In addition, owing to the fact that 1) it is rare for

multiple combinations of anchor points to produce the same pixelised target pattern and 2) the selection of anchor points obeys a uniform distribution, it can be said that each distinct target pattern has a near equal chance of being selected in every training episode. Therefore, this uniformly distributed possibility of appearance in combination with the sheer number of the possible distinct target pattern ensures that it is extremely unlikely that the reinforcement learning agent has seen these randomly generated experiment target patterns during training. And it is still more unlikely that the target patterns selected for these experiments could have occur with sufficient frequency during training that the reinforcement learning agent would be able to learn dedicated policies for XY coordination deduction for these specific target patterns. These results therefore indicate that the reinforcement learning agent has learned a generalised policy that allows it to generate toolpaths for arbitrary, randomly generated target patterns. Although it is impractical to use the CMA-ES to estimate the performance upper bound for a large number of generated target patterns, it is, however, feasible to compare these two experimental results with trajectories of XY coordinates found by the CMA-ES agent when given the same two target patterns. The CMA-ES agent successfully machines 97.36 % of the first target pattern with 28 laser pulses, but inadvertently machines 1.17 % of the target pattern area from the surrounding material; this trajectory is illustrated in Figure 7-13 d). For the second target pattern, the CMA-ES successfully machines 98.19 % of the target pattern whilst inadvertently machining 0.77 % of the target pattern area from the surrounding material using 45 laser pulses, and this trajectory is illustrated in Figure 7-13 h). Comparatively, for the first target pattern, the reinforcement learning agent machines 92.47 % of the target pattern, but inadvertently machines 3.32 % of target pattern area from the surrounding material using 33 pulses (this trajectory is shown in Figure 7-13 c)); and for the second target pattern, the reinforcement learning agent successfully machines 93.05 % of the target pattern whilst inadvertently machining 3.24 % of the target pattern area from the surrounding material using 39 laser pulses (this trajectory is shown in Figure 7-13 g).

The results reported above show a larger discrepancy in performance between the reinforcement learning agent and the CMA-ES agent, comparing to the previously reported results from the virtual environments with a fixed target pattern and with the rotating target pattern. The origins of this reduced performance are twofold:

1) Unlike in environments where the same target pattern can be seen repeatedly during the training (i.e., the fixed five-pointed star and the rotating five-pointed star) where a dedicated policy of XY coordinates can be learned for each target pattern, the reinforcement learning in this environment has only seen a small fraction of all possible target patterns ($\sim {1 \times 10^7}/{1.09 \times 10^{23}}$). Therefore, the reinforcement learning agent has

to rely purely on the learned general principle of XY coordinate deduction to perform tool-path generation for previously unseen target patterns.

2) The trajectories of XY coordinates obtained from the CMA-ES optimisation are simulation results, while the trajectories of XY coordinates obtained from the reinforcement learning agent are experimental results. This means that only the results for the reinforcement learning agent are subject to software and hardware induced experimental errors.

As per the discussion in Section 7.5, the results for the CMA-ES agent allow us to estimate the discrepancy between the results for the reinforcement learning agent and the optimum solution. The CMA-ES agent is fundamentally different from the reinforcement learning agent, in the sense that each instance of CMA-ES optimisation takes 15 minutes to complete and is only able to generate a toolpath for one target pattern, whereas the reinforcement learning agent, once trained, can deduce an entire trajectory of XY coordinate in less than a second and is applicable to a vast number of arbitrary target patterns.

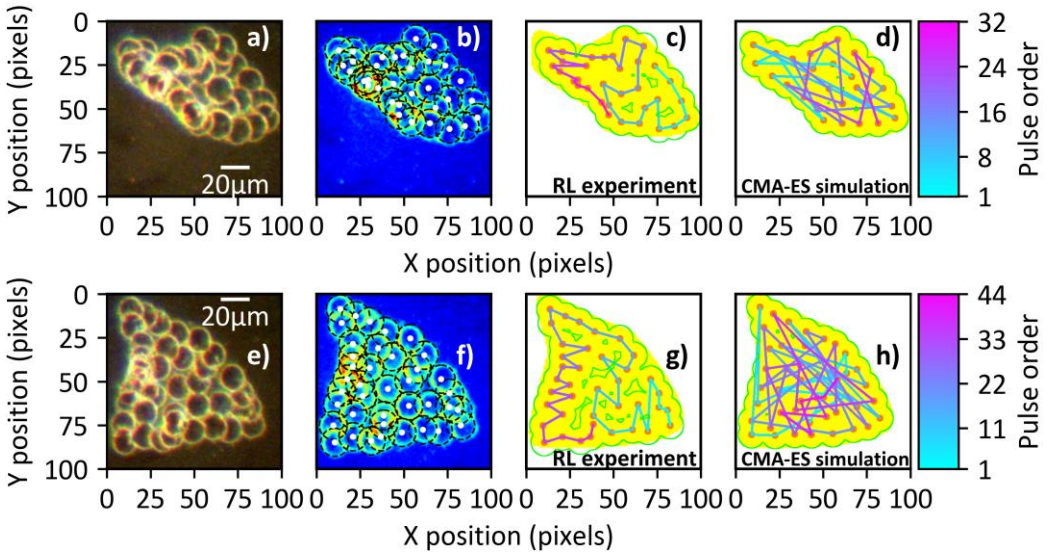

Figure 7-13 Results for two different target patterns (one on each row) showing, respectively, in columns from the left: experimental camera observations, image processing results, trajectories selected by the RL agent and trajectories selected by the CMA-ES algorithm.

7.9 Comparison with Conventional Raster Scanning

In laser materials processing, arguably the simplest, and presumably the most common method to generate a toolpath in accordance with a target pattern is the raster scanning method (Tseng et al., 2012, Prakash and Kumar, 2018, Childs and Hauser, 2005). In a raster scanning method, a target pattern is first digitised into an array of coordinates. Within this array, some coordinates are labelled as positions at which pulses should be applied (i.e., 'laser-machined pixel'), and the other

coordinates are labelled as positions where incident pulses are not permitted (i.e., 'non-machined pixel'). The criteria for determining whether a pulse should or should not be applied at a coordinate depends on various factors, such as the area of the target pattern that would be machined if a pulse is incident at that coordinate, compared to the area of the surrounding material that might be machined. In common practice, a beam positioning hardware usually scans across a workpiece row by row in accordance with the array of coordinates; when a coordinate labelled as a 'laser machined pixel' is visited, a laser pulse is applied at the workpiece, and vice versa.

Two configurations of simulated raster scanning examples are demonstrated in Figure 7-14 a) and b), where the chosen target pattern is randomly generated. The first configuration shown in Figure 7-14 a) possesses a distance between two adjacent coordinates which equals $\sqrt{2}$ times the radius of the circular laser-machined structure. This is the minimum separation which ensures that gaps (i.e., unprocessed areas) do not exist between adjacent laser-machined structures. Additionally, this separation also induces the least overlapping between two adjacent laser-machined structures. However, in this configuration, it is evident from the figure that the exterior of the target pattern can only be coarsely followed by the laser-machined structures, as that the separation between the laser-machined structures is too large for any thin feature to be appropriately machined. The trajectory of XY coordinates for the first configuration covers 82.04 % of the target pattern with 19 incident pulses, but it also inadvertently covers 2.16 % of the target pattern area from the surrounding material. Figure 7-14 b) shows the second configuration of the raster scanning simulation, where coordinates are more closed arranged. In this configuration, the separation between two adjacent coordinates equals to the radius of a circular laser-machined structure. This smaller separation causes more overlapping between adjacent laser-machined structures, but simultaneously enables a more refined machining of the perimeter of the target pattern. The trajectory of XY coordinates for the second configuration covers 89.32 % of the target pattern with 33 incident pulses, but it also inadvertently covers 1.51 % of the target pattern area from the surrounding material.

Figure 7-14 c) is the simulated result, on the same target pattern, for the reinforcement learning agent (obtained in Section 7.8). It clearly shows that the reinforcement learning agent distributes pulses more densely around the exterior contour of the target pattern in order to better follow the perimeter of the target pattern, and fills the interior of the target pattern with relatively fewer, less-overlapping pulses. The reinforcement learning agent uses 30 pulses to covers 97.21 % of the target area, whilst 5.51 % of the area of the target pattern is inadvertently covered from the surrounding material. Compared to the raster scanning method, the reinforcement learning agent is able to adaptively adjust the separations between adjacent laser-machined structures, such that

the exterior of the target pattern can be closely replicated on the workpiece, while the interior of the merged shape is machined with fewer, and less overlapped laser pulses.

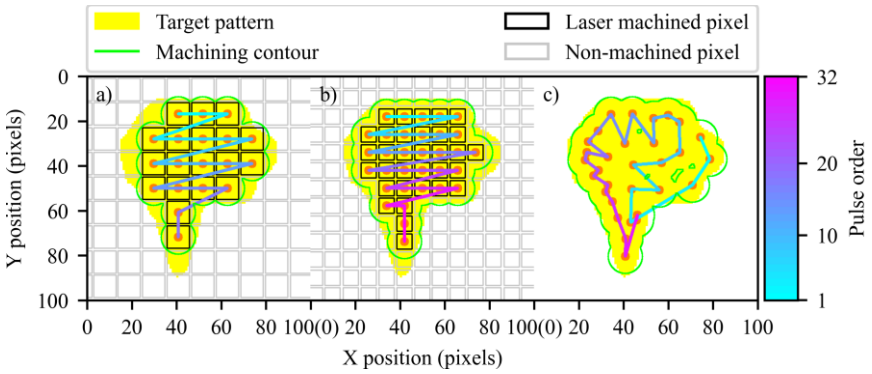

Figure 7-14 Comparison between the results for the raster scanning method with two different grid spacing a) and b). c) The result for the reinforcement learning agent approach with the same arbitrary target pattern.

7.10 A Self-correcting Ability

The adopted reinforcement learning algorithm (i.e., PPO) belongs to a greater class of reinforcement learning algorithms commonly known as Policy Gradient algorithms, which usually employ a conditional probability distribution to enable action deduction in a continuous domain. In Policy Gradient algorithms, a policy is usually written in the form of a conditional probability distribution: $Pr\{a_t = a|s_t = s, \theta\}$, and it reads as: at time step t, given a state s, the probability of deducing action a is given by $\pi(a|s)$ which is parameterised by θ. This probability distribution can be Gaussian, Beta and Bernoulli etc., and the parameterisation θ varies accordingly with the choice of the probability distribution; for instance, in a Gaussian distributed policy, the mean μ and the variance σ^2 are parameterised, and in a Beta distributed policy, the α and the β are parameterised.

The reinforcement learning agents that were trained to adaptively select XY coordinates in accordance with fixed, rotating and arbitrary target patterns all employed a Gaussian distributed policy. Specifically, in order to deduce X and Y coordinates independently, an isotropic multivariate Gaussian distribution was adopted (i.e., $\mathcal{N}(\mu, \sigma)$ where $\mu \in \mathbb{R}^2$, $\sigma \in \mathbb{R}^{2\times2}$ and $\sigma = nI$, where $\mu \in \mathbb{R}$). In the training, the mean μ is a learnable parameter, owing to the fact that it determines the most likely deduced action. Note that the mean determining the most likely action, is only true in symmetric probability distributions (e.g., Gaussian), where the mean of the distribution coincides with the mode of the distribution (i.e., the global maxima of its probability density function), whereas in a probability distribution that can be asymmetric (e.g., Beta), it is the mode of the probability distribution that determines the most likely action. On the other hand, the variance σ^2, which can be interpreted as a parameter that controls the uncertainty of a deduced action, can be

learnable or annealed, but is regardless gradually decreased as the training progresses, owing to the fact that, in many problems, the optimum policy is a deterministic policy (i.e., actions are deduction without uncertainty). Note that there exist problems where the optimum policy cannot be deterministic, for instance, rock-paper-scissors, but these problems are outside of the scope of this thesis. This is the reason that, in Section 7.6, deterministic trajectories of XY coordinates can be obtained by taking the mean μ of the Gaussian distributed policy. This is also the reason that, in Section 7.6, the reinforcement learning agent is able to compensate for inadvertent errors in laser pulse positioning that stem from software and hardware limitations. During training, the reinforcement learning agent does not always select the most likely action, it is in fact possible for any coordinate on the observable workpiece to be selected by the reinforcement learning agent, regardless of the mean and the variance, because the domain of the probability density function of a Gaussian distribution that is not truncated and extends to $\pm\infty$. The reinforcement learning agent has to learn to determine actions despite being subject to this perturbation (i.e., it has to learn to tolerate selection of non-preferred actions) whilst still maximising the discounted episodic reward. In other words, during training, a reinforcement learning agent not only optimises one trajectory of XY coordinates (with reference to a predefined target pattern), but it also explores a large number of sub-optimum trajectories where the XY positions from the optimal trajectory are perturbed. In the physical experiment of Figure 7-7, any inadvertent laser pulse positioning errors caused by software or hardware limitations can also be regarded as a perturbation that is applied to the deduced action; the reinforcement learning can, therefore, simply continue along one of these sub-optimal trajectories in the event that an erroneous action occurs.

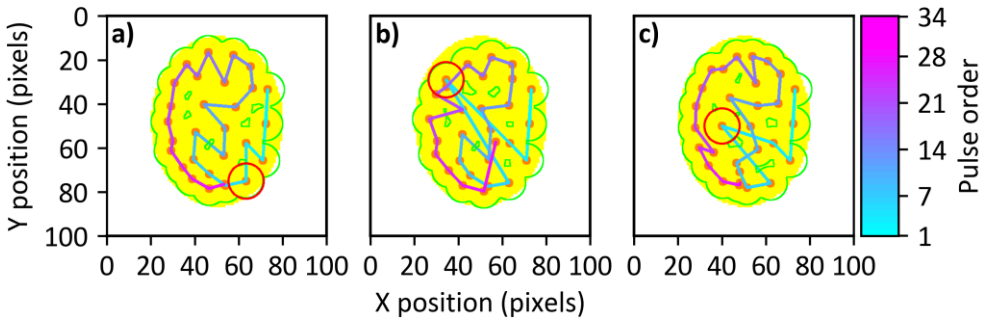

Figure 7-15 a) Simulated trajectory without any pertubation, where the 5th incident pulse is marked as red circle. b) and c) Simulated tracjectories where the 5th incident pulses are relocated away from the associated preferred position of the reinforcement learning agent.

Figure 7-15 presents a demonstration of this self-correction ability of the reinforcement learning agent. In this demonstration, the target pattern is an elliptical shape, and the decision-making agent is the reinforcement leaning agent obtained from the arbitrary target pattern environment in Section 7.8. In this figure, the simulated laser-machined structure of the 5th incident

pulse is highlighted in red in a). This 5th incident pulse is deliberately relocated to two different coordinates on the observable workpiece in Figure 7-15 b) and Figure 7-15 c). It can be seen that these perturbations do not affect the ability of the reinforcement learning agent to complete the machining in accordance with the elliptical target pattern. As a matter of fact, in Figure 7-16, the percentages of the target pattern machined when the 5th incident pulse is relocated to all possible coordinates on the observable workpiece are shown in the form of a heat map. It can be seen that the completion of machining of the target is generally not influenced by the coordinate to which the 5th incident pulse is relocated. Note that because the 5th pulse is relocated to every possible coordinate, which includes coordinates that would normally immediately terminate the associated episode, the termination check at the 5th incident pulse during this simulation is deliberately omitted. (For all pulses other than the 5th one, the bad exit condition termination check continues to work as per the description in Section 7.4).

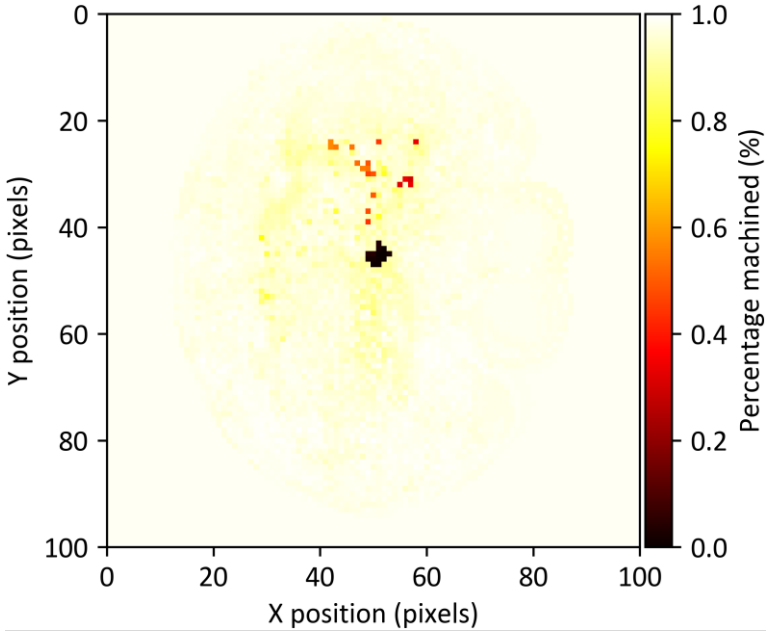

Figure 7-16 Heat map showing the percentage of the target pattern machined versus the position to which the 5th incident pulse is relocated.

The reinforcement learning agent deduces actions adaptively with reference to the observation of the workpiece. These stepwise observations and the scheme of action deduction that is of random nature, in combination, give rise to the ability of the reinforcement learning agent to detect and consequently compensate for any inadvertent incident pulse. A similar ability is in principle possible to achieve via conventional means with the CMA-ES algorithm — a monitoring system would be needed, to observe the workpiece, and detect machining errors in real time, and in the event that a machining error occurs, and leads to an unintentional laser-machined structure, a new target pattern would need to be generated, and thereafter inputted to the CMA-ES agent for a new instance of optimisation. It can be observed that in this system the self-correction ability is

not inherent within the CMA-ES agent, and thus the time taken for this process to finish machining is directly related to the number of detected unintentional laser-machined structures (since each unintentional laser-machined structure leads to a separate instance of CMA-ES optimisation for generating the subsequent trajectory of XY coordinates). On the contrary, the time taken for the reinforcement learning agent to finish the same machining process is not related to the number of detected unintentional laser-machined structures, owing to the fact that each observation of the workpiece is processed by the same numerical operations in the function approximator of the reinforcement learning agent (i.e., the neural network).

7.11 Conclusion

In conclusion, reinforcement learning has been demonstrated to be capable of toolpath generation in accordance with a predefined target pattern. This toolpath controls XY stages, upon which the workpiece is mounted, to apply incident laser pulses to the workpiece in a stepwise manner and at the XY coordinates determined by the reinforcement learning agent. Furthermore, these XY positions are based partly on camera observations that are made of the workpiece, which allows the reinforcement learning agent to monitor laser pulse positioning to identify any machining errors as they occur, in real-time. The reinforcement learning approach has been tested upon problems with increasing complexity; initially with a fixed target pattern, then with target patterns that may be rotated about their centre to an arbitrary angle, then finally with target patterns that are randomly generated for each episode and almost completely arbitrary in nature. Lastly, it has been demonstrated that the reinforcement learning agent is capable of adaptively selecting laser machining positions on the workpiece that compensate for pulses inadvertently applied at the incorrect location; this enables the agent to be robust and complete the machining of a predefined target pattern despite the presence of deliberately introduced error

Chapter 8 Conclusions and Future Work

The results in this thesis clearly demonstrate that deep learning has an important role to play in the field of photonics, for both academic research and industrial applications. Deep learning offers the capability to model, control and optimise complex physical systems, purely from the processing of experimental training data. As such, it unlocks the potential for solving physical systems that are currently too complex to solve via existing theoretical approaches using available computing power. However, whilst there is significant potential for the application of deep learning in photonics (and indeed almost every other research discipline), a major challenge is the limited degree of expertise of deep learning outside traditional computer science research fields. This thesis, which contains introduction chapters on both photonics and deep learning, therefore contributes to the development of interdisciplinary expertise across photonics and deep learning. Understandably, the interface of photonics and deep learning is only just starting to be explored, and indeed the four distinct novel experimental applications presented in this thesis provide a varied set of preliminary studies for this interface.

In this final chapter, experimental results that are presented in this thesis are briefly described and summarised, followed by a discussion of the potential for future work in this area.

8.1 Conclusions

In Chapter 4 and Chapter 5, two experiment were conducted in order to demonstrate that convolutional neural networks were able to map real-world sensory signals about a physical process to interpretable numerical descriptions of the underlying process. In Chapter 4, the scattering patterns when microparticles were illuminated by a laser were mapped to the numerical values that described the concentration of the microparticles and the salinity of the solution. The convolutional neural network was hosted in a compact single-board computer and was able to determine the aforementioned attributes accurately and precisely with the coefficient of determination in excess of 0.99. In Chapter 5, the camera observations of laser-machined structures were used to determine the translation and rotation of the laser beam with respect to a reference position. The accuracies for determining the translation of a laser-machined structure along X- and Y-axis were approximately 0.28 μm and 0.22 μm (measured in mean square error), which were both smaller than the scale of a single camera pixel. The accuracy for determining the rotation of a laser-machined structure was approximately 7 degrees. Specifically, in Chapter 5, it was demonstrated that data augmentation could be used to mimic real-world translations of the laser-machined structures, and therefore reduce the reliance on real-world data collection.

Also, in Chapter 5, the convolutional neural network was used to determine the number of pulses required for penetrating a thin copper film (with a varying thickness) that was deposited on a silica substrate, without damaging the silica substrate, based on the camera observation of the laser-machined structure. This convolutional neural network was then incorporated into a controller that was connected to both the laser source and the observing camera in order for the controller to cease laser machining in real time at the time point where the thin copper film had been completely penetrated. The convolutional neural network was approximately 28 % more accurate in determining the number of pulses required to penetrate the thin copper film, compared to that of the naïve guess based on the mean of the ground truth. In this experiment, the ground truth regarding the depth of the laser-machined structures was determined through human inspection of surface profilometer data, and was provided to the convolutional neural network during training. In many real-world control problems, it cannot be guaranteed that the ground truth regarding the optimal course of action can be obtained via human inspection, and therefore a learning paradigm that is able to determine the best course of action through interacting with an external environment is of particular interest.

In Chapter 6 and Chapter 7, reinforcement learning agents were deployed in virtual environments that approximated a physical environment, in order to master specific capabilities. The trained reinforcement learning agents were then able to be seamlessly employed in the physical environment without the need for additional training. Specifically, in Chapter 6, the reinforcement learning agent was tasked to maneuver a laser-trapped microparticle via the optical tweezers effect to a predefined target position, whilst also avoiding collisions with another free-moving microparticles that were in the same environment. It was shown that the trained reinforcement learning agent was able to perform simultaneously both the path-planning and the collision avoidance ability in both the virtual and physical environment. Additionally, it was shown that the reinforcement learning agent could be deployed in a hybrid environment where observations of the virtual environment were overlaid with the observations of the physical environment. In Chapter 7, the reinforcement learning agent was tasked to machine a glass slide in accordance with a predefined target pattern. It was shown that a single reinforcement learning agent was able to generate toolpath and control the laser machining experimental setup, in order to machine a wide range of previously unseen target patterns. In addition, it was shown that through parameterising the actuator of the reinforcement learning agent via a probability distribution, the reinforcement learning agent was able to adaptively change toolpath to compensate for any inadvertent incident pulse.

8.2 Future Work

The interface of deep learning and photonics is new and exciting, and there is clearly a wealth of novel applications to be explored. This thesis has concentrated on lasers, for machining, sensing and sample manipulation (via the optical tweezers effect), along with both supervised and unsupervised deep learning techniques. At this point in time, almost any combination of a photonics experiment and a deep learning approach is a novel exploration, and hence there is almost unlimited scope for future work. However, whilst neural networks are indeed proven to be extremely capable at modelling and solving complex systems, they are in many ways a 'black box', and therefore future research should have a particular emphasis on the development of methods for extracting useful and novel understanding from the trained neural networks. Whilst in general each of the four experimental chapters presented in this thesis represent a complete piece of work, there are some further developments to Chapter 7 that are encouraged by the author.

In Chapter 7, the studies regarding the precise control of the positions at which each incident laser pulse interacted with the workpiece were limited to the XY plane that was orthogonal to the direction of the incident laser pulses. It is perhaps of greater interest to also explore the ability of the reinforcement learning to also control the light-matter interactions in the Z-axis (along the direction of the incident laser pulses). In the discussion of Section 7.4, where the reward function is theoretically justified, this reward function can be easily apply to three-dimensional cases through altering $\mathbf{C} = \{c^b, c^t\}$ in $\mathcal{O} = \mathbf{C}^{m \times n}$ to $\mathbf{C} = \{c | 0 \leq c \leq h, c \in \mathbb{R}\}$ where c is the surface profile at a pixel (constrained by a maximum thickness of the workpiece h). In a three-dimensional case, the light-matter interactions between the incident laser pulses and the workpiece would likely be more complex than that of the demonstrated two-dimensional case, where factors such as the incubation effect and the dependency of the ablation threshold on the thickness of the workpiece might also need be considered. However, if this level of additional complexity could be solved, there would be the potential for the precise laser machining of complex three-dimensional structures.

Appendix A1 Longitudinal Cavity Mode

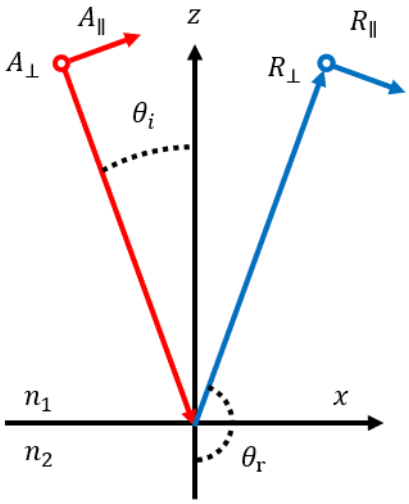

Figure A 1 Parallel and perpendicular components of the electric fields of the incident beam and the reflected beam.

Let the XY-plane $z = 0$ be the boundary between two homogeneous media; the media above has a refractive index of n_1 and the media below has a refractive index of n_2. Let the XZ-plane be the plane of incident, and the angle θ_i be the angle of incident (see Figure A 1). The real part of the electric field of the incident wave can be written as (assuming plane wave)

$$\begin{cases} E_x^i = -A_\parallel e^{-i\tau^i} \cos\theta_i \\ E_y^i = A_\perp e^{-i\tau^i} \\ E_z^i = -A_\parallel e^{-i\tau^i} \sin\theta_i \\ \tau^i = \omega\left(t - \dfrac{x\sin\theta_i - z\cos\theta_i}{c}\right) \end{cases} \quad (A1)$$

where A is the amplitude of the electric field of the incident beam, and $A_\parallel$ and $A_\perp$ are the parallel and perpendicular components of the amplitude of the electric field respectively, with respect to the plane of incident. The reflected electric field is governed by the Fresnel formula, which is given by

$$\begin{cases} R_\parallel = \dfrac{n_2\cos\theta_i - n_1\cos\theta_t}{n_2\cos\theta_i + n_1\cos\theta_t} A_\parallel \\ R_\perp = \dfrac{n_1\cos\theta_i - n_2\cos\theta_t}{n_1\cos\theta_i + n_2\cos\theta_t} A_\perp \end{cases} \quad (A2)$$

Knowing that, for optical resonators, the reflectivity of the mirrors is usually very high in order to reduce round-trip loss (ignoring output couplers), and therefore $n_2 \gg n_1$, thus equation A2 can be re-written into

$$\begin{cases} R_{\parallel} \sim A_{\parallel} \\ R_{\perp} \sim -A_{\perp} \end{cases} \tag{A3}$$

From equation A3, the reflected beam can then be described as

$$\begin{cases} E_x^r = -R_{\parallel} e^{-i\tau^r} \cos\theta_r = A_{\parallel} e^{-i\tau^r} \cos\theta_i \\ E_y^r = R_{\perp} e^{-i\tau^r} = -A_{\perp} e^{-i\tau^r} \\ E_z^r = -R_{\parallel} e^{-i\tau^r} \sin\theta_r = -A_{\parallel} e^{-i\tau^r} \sin\theta_i \\ \tau^r = \omega\left(t - \dfrac{x\sin\theta_i + z\cos\theta_i}{c}\right) \end{cases} \tag{A4}$$

and the interference between the incident and the reflected electric fields is therefore given by combining components of the incident and reflected electric fields (i.e., A1+A4)

$$\begin{cases} E_x = E_x^i + E_x^r = 2A_{\parallel}\cos\theta_i \sin(\dfrac{\omega z\cos\theta_i}{c}) \exp\left\{-i\left[\omega\left(t - \dfrac{x\sin\theta_i}{c}\right) - \dfrac{\pi}{2}\right]\right\} \\ E_y = E_y^i + E_y^r = -2A_{\perp}\sin(\dfrac{\omega z\cos\theta_i}{c}) \exp\left\{-i\left[\omega\left(t - \dfrac{x\sin\theta_i}{c}\right) - \dfrac{\pi}{2}\right]\right\} \\ E_z = E_z^i + E_z^r = -2A_{\parallel}\sin\theta_i \sin(\dfrac{\omega z\cos\theta_i}{c}) \exp\left\{-i\left[\omega\left(t - \dfrac{x\sin\theta_i}{c}\right)\right]\right\} \end{cases} \tag{A5}$$

now if the angle of incident is assumed to be zero, i.e., $\theta_i = 0$, A5 can be re-written into

$$\begin{cases} E_x = E_x^i + E_x^r = 2A_{\parallel}\sin(\dfrac{\omega z}{c}) \exp\left[-i(\omega t - \dfrac{\pi}{2})\right] \\ E_y = E_y^i + E_y^r = -2A_{\perp}\sin(\dfrac{\omega z}{c}) \exp\left[-i(\omega t - \dfrac{\pi}{2})\right] \\ E_z = 0 \end{cases} \tag{A6}$$

It can be seen from equation *A6* that when $z = \frac{c}{\omega}\pi q, q = 0, 1, 2, 3, \cdots$ the amplitude of the super-positioned electric field is always zero, and when $z = \frac{c}{\omega}\cdot\frac{\pi}{2}q, q = 1,2,3, \cdots$ the amplitude of the super-positioned electric field is always at the maxima or at the minima. If we consider z to be a fixed value (i.e., alter it with the length of a plane-parallel cavity L), the angular frequencies of the supported electric fields can be thus characterised by $\omega = \frac{\pi c}{L}\cdot q, q = 1,2,3, \cdots$.

Appendix A2 Mode-locked Longitudinal Mode Interference

Consider all longitudinal modes along x-axis from equation A6 to have a unity magnitude:

$$E(\omega) = E_x = \sin\left(\frac{\omega z}{c}\right)\exp\left[-i\left(\omega t - \frac{\pi}{2}\right)\right] \quad (A7)$$

The super-position of N consecutive longitudinal modes $\mathrm{E}(\omega_q, \Delta\omega, N)$, which starts ay $E(\omega_q)$ and is spaced by $\Delta\omega$, i.e., $E(\omega_q) + E(\omega_{q+1}) + \cdots + E(\omega_{q+N-1})$ can be written as

$$\begin{aligned}
\mathrm{E}(\omega_q, \Delta\omega, N) &= E(\omega_q) + E(\omega_{q+1}) + \cdots + E(\omega_{q+N-1}) \\
&= \sum_{n=1}^{N} \exp\left\{-i\left[(\omega_q + (n-1)\Delta\omega)t - \frac{\pi}{2}\right]\right\} \\
&= \sum_{n=1}^{N} \exp\left\{-i\left(\omega_q t - \frac{\pi}{2}\right) + [-i(n-1)\Delta\omega t]\right\} \\
&= \exp\left[-i\left(\omega_q t - \frac{\pi}{2}\right)\right]\sum_{n=1}^{N} \exp[-i(n-1)\Delta\omega t] \\
&= \exp\left[-i\left(\omega_q t - \frac{\pi}{2}\right)\right]\left(1 + e^{-i\Delta\omega t} + e^{-i2\Delta\omega t} + \cdots + + e^{-i(N-1)\Delta\omega t}\right) \\
&= \exp\left[-i\left(\omega_q t - \frac{\pi}{2}\right)\right]\left(\frac{1 - e^{-iN\Delta\omega t}}{1 - e^{-i\Delta\omega t}}\right) \\
&= \exp\left[-i\left(\omega_q t - \frac{\pi}{2}\right)\right]\left(\frac{e^{-iN\Delta\omega t/2}}{e^{-i\Delta\omega t/2}}\frac{\sin\left(N\Delta\omega t/2\right)}{\sin\left(\Delta\omega t/2\right)}\right) \\
&= \exp\left[-i\left(\omega_q t - \frac{\pi}{2}\right)\right]\exp\left[-i(N-1)\Delta\omega t/2\right]\left(\frac{\sin\left(N\Delta\omega t/2\right)}{\sin\left(\Delta\omega t/2\right)}\right) \\
&= \exp\left[-i\left(\left[\omega_q + \frac{(N-1)\Delta\omega}{2}\right]t - \frac{\pi}{2}\right)\right]\left(\frac{\sin\left(N\Delta\omega t/2\right)}{\sin\left(\Delta\omega t/2\right)}\right) \quad (A8)
\end{aligned}$$

If $\omega_q \gg (N-1)\Delta\omega/2$, A8 can be approximated by

$$\mathrm{E}(\omega_q, \Delta\omega, N) \sim \exp\left[-i\left(\omega_q t - \frac{\pi}{2}\right)\right]\left(\frac{\sin\left(N\Delta\omega t/2\right)}{\sin\left(\Delta\omega t/2\right)}\right) \quad (A9)$$

Appendix A3 Focal Length of a Kerr Lens

Considering a collimated Gaussian beam, the width radius of which is w_0. Its transverse intensity profile can be written as

$$I(r) = I_0 \exp\left(\frac{-2r^2}{{w_0}^2}\right) \quad (A10)$$

The phase shift $\Delta\varphi$ that this beam experiences, when this beam is incident on a plate of Kerr effect (i.e., $n = n_0 + n_2 I$) material with a uniform thickness d, is (assuming normal incident)

$$\Delta\varphi(r) = knd = kn_0 d + kn_2 dI_0 \exp\left(\frac{-2r^2}{{w_0}^2}\right) = kn_0 d + \varphi_0 \exp\left(\frac{-2r^2}{{w_0}^2}\right), \varphi_0 = \frac{kn_2 dP}{\lambda {w_0}^2} \quad (A11)$$

where $k = \frac{2\pi}{\lambda}$ is the wavenumber of the incident Gaussian beam, and nd is the optical path length. It can be seen from *A11* that the phase shift is a function of both the optical power P and the axial distance from the centre of the Gaussian beam r.

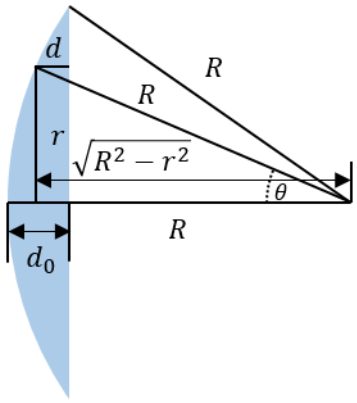

Figure A 2 Optical path difference introduced by a concave lens.

In a conventional converging lens (whose refractive index is n, see Figure A 2), the phase shift along the axial direction is

$$\begin{aligned}\Delta\varphi(r) &= k[nd + (d_0 - d)] \\ &= k[(n-1)d + d_0]\end{aligned} \quad (A12)$$

With the paraxial approximation ($\sin\theta = \frac{r}{R} \sim \theta$, $\cos\theta = \frac{\sqrt{R^2 - r^2}}{R} \sim 1 - \frac{\theta^2}{2}$) and lensmaker's equation ($\frac{1}{f} = \frac{n-1}{R}$), *A12* can be approximated as

$$\Delta\varphi(r) = k\left\{(n-1)\left[d_0 - \left(R - \sqrt{R^2 - r^2}\right)\right] + d_0\right\}$$

$$= k\left[(n-1)\left(d_0 - \frac{r^2}{2R}\right) + d_0\right] = knd_0 - \frac{kr^2}{2f} \qquad (A13)$$

It can thus be seen from *A11* and *A13* that, for both the Kerr lens and the conventional converging lens, the phase change decreases with the axial distance r. Additionally, for the Kerr lens, the phase shift also depends on the optical power P. To estimate the focal distance f for the Kerr lens requires further approximation of equation *A11*, which is available in reference [Placeholder].

Appendix A4 Optical Path Difference Introduced by DMD Mirrors

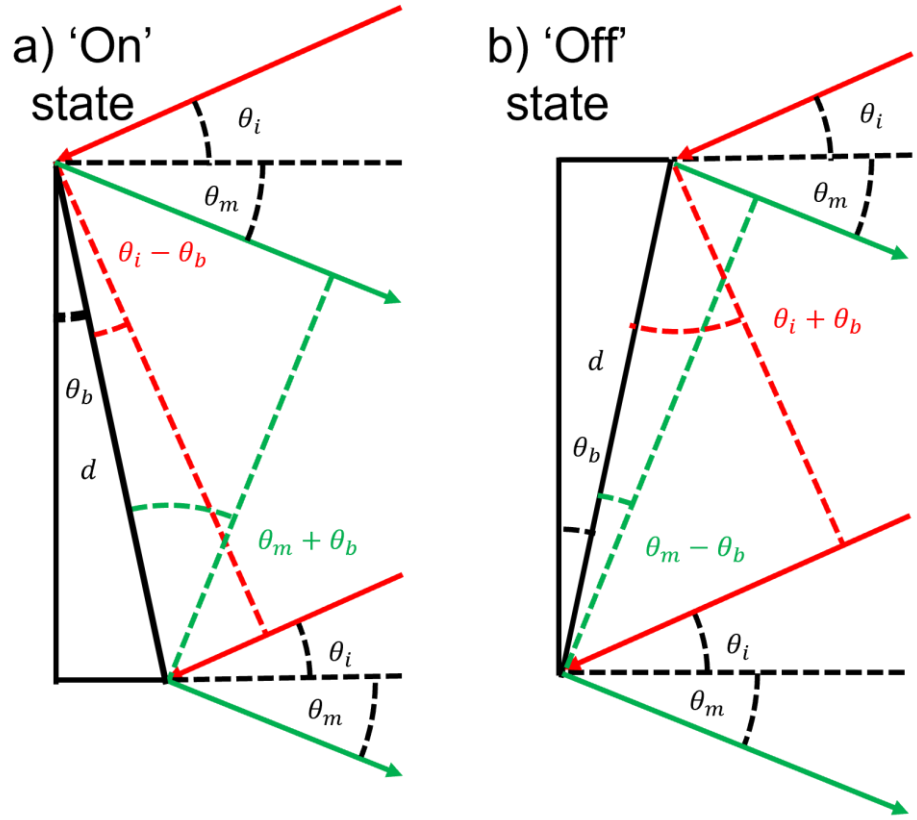

Figure A 3 Optical path difference introduced by a single-slit.

The schematics of the optical path differences that are introduced by a single DMD mirror that is in the 'On' state and 'Off' state is shown in Figure A 3 a) and b) respectively; the mathematical expressions of these optical path differences in 'On' state and in 'Off' state can thus be written as

$$\delta_{on} = d[\cos(\theta_i - \theta_b) - \cos(\theta_m + \theta_d)] \quad A14$$

$$\delta_{off} = d[\cos(\theta_i + \theta_b) - \cos(\theta_m - \theta_d)] \quad A15$$

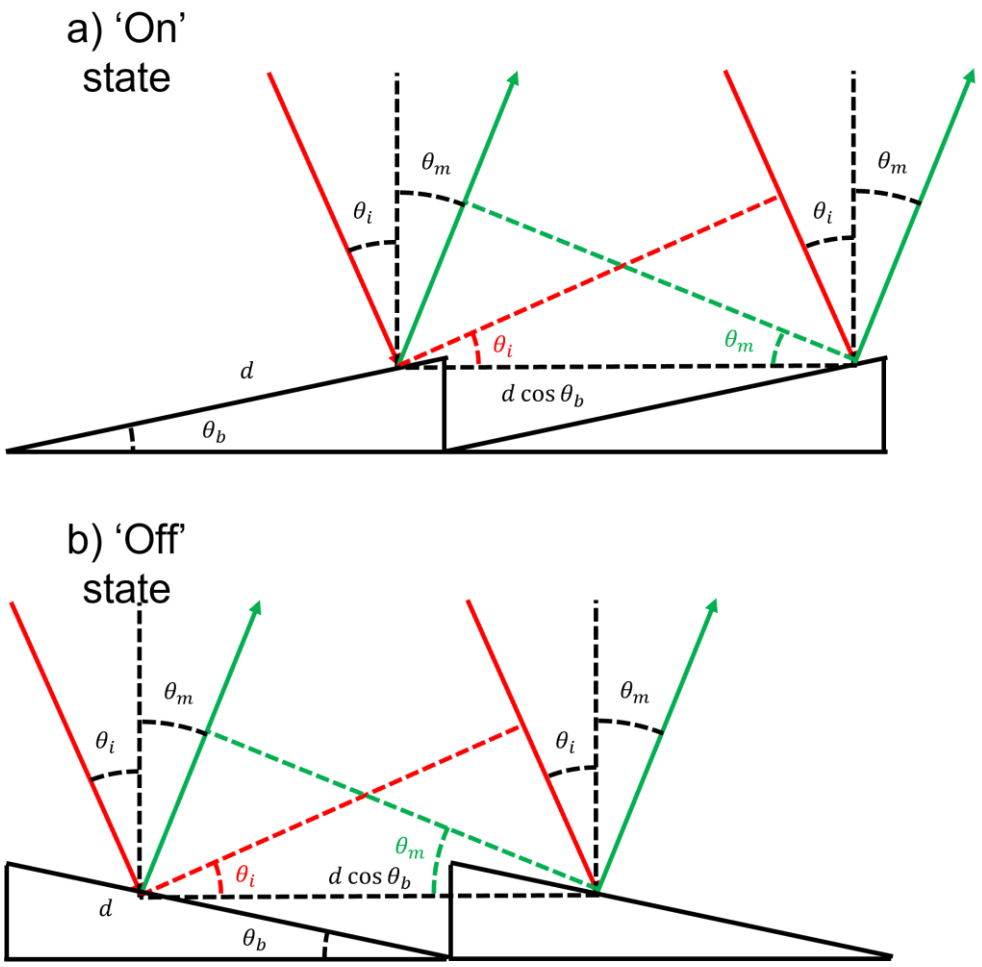

Figure A 4 Optical path difference introduced by multiple slits.

The schematics of the optical path differences that are introduced by two DMD mirrors that are both in the 'On' state and 'Off' state is shown in Figure A 4 a) and b) respectively; the mathematical expressions of these optical path differences in 'On' state and in 'Off' state are identical, with no dependence on the 'On' and 'Off' state of the DMD mirrors, and it reads

$$\delta_{multiple} = d\cos\theta_b\,[\sin\theta_i - \sin\theta_m] \qquad (A14)$$

Note that differs from the common practice that set the normal to be orthogonal to the tilted surface of a DMD mirror (i.e., facet normal), the normal in both Figure A 3 and Figure A 5 is set to be orthogonal to the plane of the DMD array (i.e., surface normal). This change is purposely made in order to ensure that results from the 'On' state DMD mirrors can be directly compared to the results from the 'Off' state DMD mirror in the same figure without the need to convert the resultant angles with respect to the same reference.

Appendix A5 Hyperparameters

Activation	ReLU
Learning rate	0.0001
Optimiser	Adam
Weight decay	0.0005
Dropout rate	50%

Table A 1 Hyperparameters for the neural networks in Section 4.3.

Activation	ReLU
Learning rate	0.0001
Optimiser	Adam
Weight decay	0.0005

Table A 2 Hyperparameters for the neural networks in Section 5.3.

Activation	ReLU
Learning rate actor	0.0001
Learning rate critics	0.0001
Optimiser	Adam
Polyak coefficient	0.995
Memory size	1,050,000
Actor noise	0.1
Target noise	0.2
Target noise clip	0.5
Batch size	32
Policy delay factor	3

Table A 3 Hyperparameters for the neural networks in Section 6.3.

Activation	ReLU
Learning rate actor	0.00025
Learning rate critics	0.00025
Optimiser	Adam
Discount factor	0.997
GAE lambda	0.95
Number of environments	16
Batch size	2048
Mini batch size	32
Surrogate epochs	10
Clipping	0.1
Value function coefficient	0.5
Entropy coefficient	0.1
SIL update	10
SIL value	0.1
SIL alpha	0.6
SIL beta	0.1

Table A 4 Hyperparameters for the neural networks in Section 7.3

List of References

ABADI, M. TensorFlow: learning functions at scale. Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, 2016. 1-1.

ACHIAM, J. 2018. *Spinning Up in Deep Reinforcement Learning* [Online]. Available: https://github.com/openai/spinningup [Accessed].

AKHMANOV, S. A., VYSLOUKH, V. A., CHIRKIN, A. S. & ATANOV, Y. 1992. *Optics of femtosecond laser pulses*, Springer.

ALWALA, K. V. & MUKADAM, M. Joint sampling and trajectory optimization over graphs for online motion planning. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020. IEEE, 4700-4707.

ASHKIN, A. 1992. Forces of a single-beam gradient laser trap on a dielectric sphere in the ray optics regime. *Biophysical journal,* 61**,** 569-582.

AUDET, C. & HARE, W. 2017. *Derivative-free and blackbox optimization*, Springer.

BANSAL, T., BELANGER, D. & MCCALLUM, A. Ask the gru: Multi-task learning for deep text recommendations. proceedings of the 10th ACM Conference on Recommender Systems, 2016. 107-114.

BELLMAN, R. 1966. Dynamic programming. *Science,* 153**,** 34-37.

BERNER, C., BROCKMAN, G., CHAN, B., CHEUNG, V., DĘBIAK, P., DENNISON, C., FARHI, D., FISCHER, Q., HASHME, S. & HESSE, C. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

BERTSIMAS, D. & TSITSIKLIS, J. 1993. Simulated annealing. *Statistical science,* 8**,** 10-15.

BEYER, H.-G. 2001. *The theory of evolution strategies*, Springer Science & Business Media.

BOHREN, C. F. & HUFFMAN, D. R. 2008. *Absorption and scattering of light by small particles*, John Wiley & Sons.

BOJARSKI, M., DEL TESTA, D., DWORAKOWSKI, D., FIRNER, B., FLEPP, B., GOYAL, P., JACKEL, L. D., MONFORT, M., MULLER, U. & ZHANG, J. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

BOLLIG, A., ABEL, D., KRATZSCH, C. & KAIERLE, S. Identification and predictive control of laser beam welding using neural networks. 2003 European Control Conference (ECC), 2003. IEEE, 2457-2462.

BOUWMEESTER, H., HOLLMAN, P. C. & PETERS, R. J. 2015. Potential health impact of environmentally released micro-and nanoplastics in the human food production chain: experiences from nanotoxicology. *Environmental science & technology,* 49**,** 8932-8947.

BRADSKI, G. & KAEHLER, A. 2000. OpenCV. *Dr. Dobb's journal of software tools,* 3**,** 2.

BROWN, T., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J. D., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G. & ASKELL, A. 2020. Language models are few-shot learners. *Advances in neural information processing systems,* 33**,** 1877-1901.

BUCHNEV, O., GRANT-JACOB, J. A., EASON, R. W., ZHELUDEV, N. I., MILLS, B. & MACDONALD, K. F. 2022. Deep-Learning-Assisted Focused Ion Beam Nanofabrication. *Nano Letters*.

BYSKOV-NIELSEN, J., SAVOLAINEN, J.-M., CHRISTENSEN, M. S. & BALLING, P. 2010. Ultra-short pulse laser ablation of metals: threshold fluence, incubation coefficient and ablation rates. *Applied Physics A,* 101**,** 97-101.

CANOVA, F., CHAMBARET, J.-P., MOUROU, G., SENTIS, M., UTEZA, O., DELAPORTE, P., ITINA, T., NATOLI, J.-Y., COMMANDRE, M. & AMRA, C. Complete characterization of damage threshold in titanium doped sapphire crystals with nanosecond, picosecond, and femtosecond laser pulses. Laser-Induced Damage in Optical Materials: 2005, 2006. SPIE, 639-645.

CHAU, M. & CHEN, H. 2008. A machine learning approach to web page filtering using content and structure analysis. *Decision Support Systems,* 44**,** 482-494.

CHILDS, T. & HAUSER, C. 2005. Raster scan selective laser melting of the surface layer of a tool steel powder bed. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* 219**,** 379-384.

CHIMIER, B., UTéZA, O., SANNER, N., SENTIS, M., ITINA, T., LASSONDE, P., LéGARé, F., VIDAL, F. & KIEFFER, J. 2011. Damage and ablation thresholds of fused-silica in femtosecond regime. *Physical Review B,* 84**,** 094104.

CHRISTOPHER, J. 1992. Technical note q-learning. *Machine learning,* 8.

CRANCH, G. A., FLOCKHART, G. M. & KIRKENDALL, C. K. 2008. Distributed feedback fiber laser strain sensors. *IEEE Sensors Journal,* 8**,** 1161-1172.

DAHL, G. E., SAINATH, T. N. & HINTON, G. E. Improving deep neural networks for LVCSR using rectified linear units and dropout. 2013 IEEE international conference on acoustics, speech and signal processing, 2013. IEEE, 8609-8613.

DILLON, J. V., LANGMORE, I., TRAN, D., BREVDO, E., VASUDEVAN, S., MOORE, D., PATTON, B., ALEMI, A., HOFFMAN, M. & SAUROUS, R. A. 2017. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*.

DOMKE, M., NOBILE, L., RAPP, S., EISELEN, S., SOTROP, J., HUBER, H. P. & SCHMIDT, M. 2014. Understanding thin film laser ablation: The role of the effective penetration depth and the film thickness. *Physics Procedia,* 56**,** 1007-1014.

DU, S. S., WANG, Y., ZHAI, X., BALAKRISHNAN, S., SALAKHUTDINOV, R. R. & SINGH, A. 2018. How many samples are needed to estimate a convolutional neural network? *Advances in Neural Information Processing Systems,* 31.

FARABET, C., COUPRIE, C., NAJMAN, L. & LECUN, Y. 2012. Scene parsing with multiscale feature learning, purity trees, and optimal covers. *arXiv preprint arXiv:1202.2160*.

FRISVAD, J. R., CHRISTENSEN, N. J. & JENSEN, H. W. 2007. Computing the scattering properties of participating media using Lorenz-Mie theory. *ACM SIGGRAPH 2007 papers.*

FUJIMOTO, S., HOOF, H. & MEGER, D. Addressing function approximation error in actor-critic methods. International conference on machine learning, 2018. PMLR, 1587-1596.

GAMALY, E. G., RODE, A. V., LUTHER-DAVIES, B. & TIKHONCHUK, V. T. 2002. Ablation of solids by femtosecond lasers: Ablation mechanism and ablation thresholds for metals and dielectrics. *Physics of plasmas,* 9**,** 949-957.

GATTASS, R. R. & MAZUR, E. 2008. Femtosecond laser micromachining in transparent materials. *Nature photonics,* 2**,** 219-225.

GAUTHIER, R. & WALLACE, S. 1995. Optical levitation of spheres: analytical development and numerical computations of the force equations. *JOSA B,* 12**,** 1680-1686.

GENTY, G., SALMELA, L., DUDLEY, J. M., BRUNNER, D., KOKHANOVSKIY, A., KOBTSEV, S. & TURITSYN, S. K. 2021. Machine learning and applications in ultrafast photonics. *Nature Photonics,* 15**,** 91-101.

GOODFELLOW, I., BENGIO, Y. & COURVILLE, A. 2016. *Deep learning*, MIT press.

GOUESBET, G. & GRéHAN, G. 1999. Generalized Lorenz-Mie theory for assemblies of spheres and aggregates. *Journal of Optics A: Pure and Applied Optics,* 1**,** 706.

GRANT-JACOB, J. A., JAIN, S., XIE, Y., MACKAY, B. S., MCDONNELL, M. D., PRAEGER, M., LOXHAM, M., RICHARDSON, D. J., EASON, R. W. & MILLS, B. 2019a. Fibre-optic based particle sensing via deep learning. *Journal of Physics: Photonics,* 1**,** 044004.

GRANT-JACOB, J. A., MACKAY, B. S., BAKER, J. A., HEATH, D. J., XIE, Y., LOXHAM, M., EASON, R. W. & MILLS, B. 2018. Real-time particle pollution sensing using machine learning. *Optics Express,* 26**,** 27237-27246.

GRANT-JACOB, J. A., MACKAY, B. S., BAKER, J. A., XIE, Y., HEATH, D. J., LOXHAM, M., EASON, R. W. & MILLS, B. 2019b. A neural lens for super-resolution biological imaging. *Journal of Physics Communications,* 3**,** 065004.

GRANT-JACOB, J. A., PRAEGER, M., EASON, R. W. & MILLS, B. 2021a. In-flight sensing of pollen grains via laser scattering and deep learning. *Engineering Research Express,* 3**,** 025021.

GRANT-JACOB, J. A., PRAEGER, M., EASON, R. W. & MILLS, B. 2021b. Semantic segmentation of pollen grain images generated from scattering patterns via deep learning. *Journal of Physics Communications,* 5**,** 055017.

GRANT-JACOB, J. A., PRAEGER, M., LOXHAM, M., EASON, R. W. & MILLS, B. 2020. Lensless imaging of pollen grains at three-wavelengths using deep learning. *Environmental Research Communications,* 2**,** 075005.

GRANT-JACOB, J. A., XIE, Y., MACKAY, B. S., PRAEGER, M., MCDONNELL, M. D., HEATH, D. J., LOXHAM, M., EASON, R. W. & MILLS, B. 2019c. Particle and salinity sensing for the marine environment via deep learning using a Raspberry Pi. *Environmental Research Communications,* 1**,** 035001.

HADSELL, R., SERMANET, P., BEN, J., ERKAN, A., SCOFFIER, M., KAVUKCUOGLU, K., MULLER, U. & LECUN, Y. 2009. Learning long‐range vision for autonomous off‐road driving. *Journal of Field Robotics,* 26**,** 120-144.

HAHNLOSER, R. H., SARPESHKAR, R., MAHOWALD, M. A., DOUGLAS, R. J. & SEUNG, H. S. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature,* 405**,** 947-951.

HANSEN, N. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

HE, K., ZHANG, X., REN, S. & SUN, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 770-778.

HEATH, D. J., GRANT-JACOB, J. A., XIE, Y., MACKAY, B. S., BAKER, J. A., EASON, R. W. & MILLS, B. 2018. Machine learning for 3D simulated visualization of laser machining. *Optics Express,* 26**,** 21574-21584.

HIGHTOWER, R., RICHARDSON, C., LIN, H.-B., EVERSOLE, J. D. & CAMPILLO, A. J. 1988. Measurements of scattering of light from layered microspheres. *Optics letters,* 13**,** 946-948.

HILL, A. A. R., ANTONIN AND ERNESTUS, MAXIMILIAN AND GLEAVE, ADAM AND KANERVISTO, ANSSI AND TRAORE, RENE AND DHARIWAL, PRAFULLA AND HESSE, CHRISTOPHER AND KLIMOV, OLEG AND NICHOL, ALEX AND PLAPPERT, MATTHIAS AND RADFORD, ALEC AND SCHULMAN, JOHN AND SIDOR, SZYMON AND WU, YUHUAI. 2018. *Stable Baselines* [Online]. GitHub. Available: https://github.com/hill-a/stable-baselines [Accessed].

HINOJOSA-ALVARADO, A. & GUTIéRREZ-VEGA, J. C. 2010. Geometrical optics calculation of forces and torques produced by a ringed beam on a prolate spheroid. *JOSA B,* 27**,** 1651-1658.

IPPEN, E., SHANK, C. & DIENES, A. 1972. Passive mode locking of the cw dye laser. *Applied Physics Letters,* 21**,** 348-350.

JAAKKOLA, T., SINGH, S. & JORDAN, M. 1994. Reinforcement learning algorithm for partially observable Markov decision problems. *Advances in neural information processing systems,* 7.

JANDELEIT, J., URBASCH, G., HOFFMANN, H., TREUSCH, H.-G. & KREUTZ, E. 1996. Picosecond laser ablation of thin copper films. *Applied Physics A,* 63**,** 117-121.

JONO, T., TAKAYAMA, Y., KURA, N., OHINATA, K., KOYAMA, Y., SHIRATAMA, K., SODNIK, Z., DEMELENNE, B., BIRD, A. & ARAI, K. OICETS on-orbit laser communication experiments. Free-Space Laser Communication Technologies XVIII, 2006. SPIE, 13-23.

KANNATEY-ASIBU JR, E. 2009. *Principles of laser materials processing*, John Wiley & Sons.

KINGMA, D. P. & BA, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

KONDA, V. & TSITSIKLIS, J. 1999. Actor-critic algorithms. *Advances in neural information processing systems,* 12.

KRAMER, M. A. 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal,* 37**,** 233-243.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems,* 25.

LAMB JR, W. E. 1964. Theory of an optical maser. *Physical Review,* 134**,** A1429.

LECUN, Y., BENGIO, Y. & HINTON, G. 2015. Deep learning. *nature,* 521**,** 436-444.

LECUN, Y., BOTTOU, L., BENGIO, Y. & HAFFNER, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 86**,** 2278-2324.

LI, Q., LI, R., JI, K. & DAI, W. Kalman filter and its application. 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), 2015. IEEE, 74-77.

LILLICRAP, T. P., HUNT, J. J., PRITZEL, A., HEESS, N., EREZ, T., TASSA, Y., SILVER, D. & WIERSTRA, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

MA, W., LIU, Z., KUDYSHEV, Z. A., BOLTASSEVA, A., CAI, W. & LIU, Y. 2021. Deep learning for the design of photonic structures. *Nature Photonics,* 15**,** 77-90.

MACKAY, B. S., BLUNDELL, S., ETTER, O., XIE, Y., MCDONNELL, M. D., PRAEGER, M., GRANT-JACOB, J., EASON, R., LEWIS, R. M. & MILLS, B. Automated 3D Labelling of Fibroblasts and Endothelial Cells in SEM-Imaged Placenta using Deep Learning. BIOIMAGING, 2020a. 46-53.

MACKAY, B. S., MARSHALL, K., GRANT-JACOB, J. A., KANCZLER, J., EASON, R. W., OREFFO, R. O. & MILLS, B. 2021. The future of bone regeneration: integrating AI into tissue engineering. *Biomedical Physics & Engineering Express,* 7**,** 052002.

MACKAY, B. S., PRAEGER, M., GRANT-JACOB, J. A., KANCZLER, J., EASON, R. W., OREFFO, R. O. & MILLS, B. 2020b. Modeling adult skeletal stem cell response to laser-machined topographies through deep learning. *Tissue and Cell,* 67**,** 101442.

MCDONNELL, M., GRANT-JACOB, J., PRAEGER, M., EASON, R. & MILLS, B. 2021a. Identification of spatial intensity profiles from femtosecond laser machined depth profiles via neural networks. *Optics Express,* 29**,** 36469-36486.

MCDONNELL, M., GRANT-JACOB, J., XIE, Y., PRAEGER, M., MACKAY, B., EASON, R. & MILLS, B. 2020. Modelling laser machining of nickel with spatially shaped three pulse sequences using deep learning. *Optics Express,* 28**,** 14627-14637.

MCDONNELL, M. D., ARNALDO, D., PELLETIER, E., GRANT-JACOB, J. A., PRAEGER, M., KARNAKIS, D., EASON, R. W. & MILLS, B. 2021b. Machine learning for multi-dimensional optimisation and predictive visualisation of laser machining. *Journal of Intelligent Manufacturing,* 32**,** 1471-1483.

MCELROY, J. H., MCAVOY, N., JOHNSON, E., DEGNAN, J., GOODWIN, F., HENDERSON, D., NUSSMEIER, T., STOKES, L., PEYTON, B. & FLATTAU, T. 1977. CO 2 laser communication systems for near-earth space applications. *Proceedings of the IEEE,* 65**,** 221-251.

MELLE, S. M., ALAVIE, A. T., KARR, S., COROY, T., LIU, K. & MEASURES, R. M. 1993. A Bragg grating-tuned fiber laser strain sensor system. *IEEE Photonics Technology Letters,* 5**,** 263-266.

MILLS, B., GRANT-JACOB, J. A., PRAEGER, M., EASON, R. W., NILSSON, J. & ZERVAS, M. N. 2022. Single step phase optimisation for coherent beam combination using deep learning. *Scientific Reports,* 12**,** 1-12.

MILLS, B. & GRANT‐JACOB, J. A. 2021. Lasers that learn: The interface of laser machining and machine learning. *IET Optoelectronics,* 15**,** 207-224.

MILLS, B., HEATH, D. J., GRANT-JACOB, J. A. & EASON, R. W. 2018a. Predictive capabilities for laser machining via a neural network. *Optics express,* 26**,** 17245-17253.

MILLS, B., HEATH, D. J., GRANT-JACOB, J. A., XIE, Y. & EASON, R. W. 2018b. Image-based monitoring of femtosecond laser machining via a neural network. *Journal of Physics: Photonics,* 1**,** 015008.

MITCHELL, T. M. 1997. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill,* 45**,** 870-877.

MNIH, V., KAVUKCUOGLU, K., SILVER, D., RUSU, A. A., VENESS, J., BELLEMARE, M. G., GRAVES, A., RIEDMILLER, M., FIDJELAND, A. K. & OSTROVSKI, G. 2015. Human-level control through deep reinforcement learning. *nature,* 518**,** 529-533.

MORRISON, E., MEERS, B. J., ROBERTSON, D. I. & WARD, H. 1994. Experimental demonstration of an automatic alignment system for optical interferometers. *Applied optics,* 33**,** 5037-5040.

MUKADAM, M., DONG, J., YAN, X., DELLAERT, F. & BOOTS, B. 2018. Continuous-time Gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research,* 37**,** 1319-1340.

MUSTAFA, H., MEZERA, M., MATTHEWS, D. T. A. & RöMER, G. 2019. Effect of surface roughness on the ultrashort pulsed laser ablation fluence threshold of zinc and steel. *Applied surface science,* 488**,** 10-21.

NELDER, J. A. & MEAD, R. 1965. A simplex method for function minimization. *The computer journal,* 7**,** 308-313.

OSINGA, F. P. 2007. *Science, strategy and war: The strategic theory of John Boyd*, Routledge.

PENDRITH, M. D. & RYAN, M. R. 1997. *Estimator variance in reinforcement learning: Theoretical problems and practical solutions*, University of New South Wales, School of Computer Science and Engineering.

PHILIPS, L. A., RUFFNER, D. B., CHEONG, F. C., BLUSEWICZ, J. M., KASIMBEG, P., WAISI, B., MCCUTCHEON, J. R. & GRIER, D. G. 2017. Holographic characterization of contaminants in water: Differentiation of suspended particles in heterogeneous dispersions. *Water research,* 122**,** 431-439.

PRAEGER, M., XIE, Y., GRANT-JACOB, J. A., EASON, R. W. & MILLS, B. 2021. Playing optical tweezers with deep reinforcement learning: in virtual, physical and augmented environments. *Machine Learning: Science and Technology,* 2**,** 035024.

PRAHL, S. A. 2022. *miepython* [Online]. Available: https://github.com/scottprahl/miepython [Accessed 2022].

PRAKASH, S. & KUMAR, S. 2018. Pulse smearing and profile generation in CO2 laser micromachining on PMMA via raster scanning. *Journal of Manufacturing Processes,* 31**,** 116-123.

RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D. & SUTSKEVER, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog,* 1**,** 9.

RETHFELD, B., IVANOV, D. S., GARCIA, M. E. & ANISIMOV, S. I. 2017. Modelling ultrafast laser ablation. *Journal of Physics D: Applied Physics,* 50**,** 193001.

REUTZEL, E. W. & NASSAR, A. R. 2015. A survey of sensing and control systems for machine and process monitoring of directed-energy, metal-based additive manufacturing. *Rapid Prototyping Journal*.

RUMELHART, D. E., HINTON, G. E. & WILLIAMS, R. J. 1986. Learning representations by back-propagating errors. *nature,* 323**,** 533-536.

SALIN, F., SQUIER, J. & PICHé, M. 1991. Mode locking of Ti: Al 2 O 3 lasers and self-focusing: a Gaussian approximation. *Optics letters,* 16**,** 1674-1676.

SCHULMAN, J., LEVINE, S., ABBEEL, P., JORDAN, M. & MORITZ, P. Trust region policy optimization. International conference on machine learning, 2015. PMLR, 1889-1897.

SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A. & KLIMOV, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

SHASTRI, B. J., TAIT, A. N., FERREIRA DE LIMA, T., PERNICE, W. H., BHASKARAN, H., WRIGHT, C. D. & PRUCNAL, P. R. 2021. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics,* 15**,** 102-114.

SHORTEN, C. & KHOSHGOFTAAR, T. M. 2019. A survey on image data augmentation for deep learning. *Journal of big data,* 6**,** 1-48.

SILVER, D., HUBERT, T., SCHRITTWIESER, J., ANTONOGLOU, I., LAI, M., GUEZ, A., LANCTOT, M., SIFRE, L., KUMARAN, D. & GRAEPEL, T. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science,* 362**,** 1140-1144.

SILVER, D., LEVER, G., HEESS, N., DEGRIS, T., WIERSTRA, D. & RIEDMILLER, M. Deterministic policy gradient algorithms. International conference on machine learning, 2014. PMLR, 387-395.

SILVER, D., SCHRITTWIESER, J., SIMONYAN, K., ANTONOGLOU, I., HUANG, A., GUEZ, A., HUBERT, T., BAKER, L., LAI, M. & BOLTON, A. 2017. Mastering the game of go without human knowledge. *nature,* 550**,** 354-359.

SONG, L. & MAZUMDER, J. 2010. Feedback control of melt pool temperature during laser cladding process. *IEEE Transactions on control systems technology,* 19**,** 1349-1356.

SONTAG, E. D. 1998. VC dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences,* 168**,** 69-96.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. & SALAKHUTDINOV, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research,* 15**,** 1929-1958.

STOIAN, R., ASHKENASI, D., ROSENFELD, A. & CAMPBELL, E. 2000. Coulomb explosion in ultrashort pulsed laser ablation of Al 2 O 3. *Physical review B,* 62**,** 13167.

STRICKLAND, D. & MOUROU, G. 1985. Compression of amplified chirped optical pulses. *Optics communications,* 55**,** 447-449.

SUGIOKA, K. & CHENG, Y. 2012. Femtosecond laser processing for optofluidic fabrication. *Lab on a Chip,* 12**,** 3576-3589.

SUN, Z. & ION, J. 1995. Laser welding of dissimilar metal combinations. *Journal of Materials Science,* 30**,** 4205-4214.

SUTTON, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning,* 3**,** 9-44.

SUTTON, R. S. & BARTO, A. G. 2018. *Reinforcement learning: An introduction*, MIT press.

SUTTON, R. S., MCALLESTER, D., SINGH, S. & MANSOUR, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems,* 12.

SVELTO, O. & HANNA, D. C. 1998. *Principles of lasers*, Springer.

SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V. & RABINOVICH, A. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. 1-9.

TATARCHENKO, M., RICHTER, S. R., RANFTL, R., LI, Z., KOLTUN, V. & BROX, T. What do single-view 3d reconstruction networks learn? Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 3405-3414.

TERAKAWA, M. & NEDYALKOV, N. N. 2016. Near-field optics for nanoprocessing. *Advanced Optical Technologies,* 5**,** 17-28.

TSENG, M. L., WU, P. C., SUN, S., CHANG, C. M., CHEN, W. T., CHU, C. H., CHEN, P.-L., ZHOU, L., HUANG, D.-W. & YEN, T.-J. 2012. Fabrication of multilayer metamaterials by femtosecond laser‐induced forward‐transfer technique. Wiley Online Library.

VAN HASSELT, H., GUEZ, A. & SILVER, D. Deep reinforcement learning with double q-learning. Proceedings of the AAAI conference on artificial intelligence, 2016.

VINYALS, O., BABUSCHKIN, I., CZARNECKI, W. M., MATHIEU, M., DUDZIK, A., CHUNG, J., CHOI, D. H., POWELL, R., EWALDS, T. & GEORGIEV, P. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature,* 575**,** 350-354.

WEINER, A. M. 2000. Femtosecond pulse shaping using spatial light modulators. *Review of scientific instruments,* 71**,** 1929-1960.

WILLIS, D. & XU, X. 2002. In situ photography of picosecond laser ablation of nickel. *Applied surface science,* 197**,** 118-123.

XIE, Y., HEATH, D. J., GRANT-JACOB, J. A., MACKAY, B. S., MCDONNELL, M. D., PRAEGER, M., EASON, R. W. & MILLS, B. 2019. Deep learning for the monitoring and process control of femtosecond laser machining. *Journal of Physics: Photonics,* 1**,** 035002.

XING, F., XIE, Y., SU, H., LIU, F. & YANG, L. 2017. Deep learning in microscopy image analysis: A survey. *IEEE transactions on neural networks and learning systems,* 29**,** 4550-4568.

YAO, K., UNNI, R. & ZHENG, Y. 2019. Intelligent nanophotonics: merging photonics and artificial intelligence at the nanoscale. *Nanophotonics,* 8**,** 339-366.

YATOOSHI, T., ISHIKAWA, A. & TSURUTA, K. 2015. Terahertz wavefront control by tunable metasurface made of graphene ribbons. *Applied Physics Letters,* 107**,** 053105.

YODER, L. A., DUNCAN, W. M., KOONTZ, E. M., SO, J., BARTLETT, T. A., LEE, B. L., SAWYERS, B. D., POWELL, D. & RANCURET, P. DLP technolgy: applications in optical networking. Spatial light modulators: Technology and applications, 2001. International Society for Optics and Photonics, 54-61.

YOU, D., GAO, X. & KATAYAMA, S. 2014. Review of laser welding monitoring. *Science and technology of welding and joining,* 19**,** 181-201.

ŽEMAITIS, A., GAIDYS, M., GEČYS, P., RAČIUKAITIS, G. & GEDVILAS, M. 2019. Rapid high-quality 3D micro-machining by optimised efficient ultrashort laser ablation. *Optics and Lasers in Engineering,* 114**,** 83-89.

ZHAO, W., QUERALTA, J. P. & WESTERLUND, T. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2020. IEEE, 737-744.

ZITOVA, B. & FLUSSER, J. 2003. Image registration methods: a survey. *Image and vision computing,* 21**,** 977-1000.