Information embedding and retrieval in 3D printed objects

ZHANG, XIN

Information embedding and retrieval in 3D printed objects

Xin Zhang

A Thesis presented for the degree of
Doctor of Philosophy

Department of Computer Sciences
University of Durham
UK

August 2020

Declaration

This thesis is submitted to the Durham University in partial fulfilment of the requirements for admission to the degree of Doctor of Philosophy. The work presented here is my own, except where expressly stated otherwise, and was performed in the Department of Computer Science at the Durham University under the supervisor of Associate Professor Ioannis Ivrissimtzis during the period July 2017 to August 2020. The research materials have not been submitted, either in the same or different form, to this or any other university for a degree. All sources of information are explicitly acknowledged.

Information embedding and retrieval in 3D printed objects

Xin Zhang

Submitted to the Durham University for the degree of Doctor of Philosophy
August 2020

Abstract

Deep learning and convolutional neural networks have become the main tools of computer vision. These techniques are good at using supervised learning to learn complex representations from data. In particular, under limited settings, the image recognition model now performs better than the human baseline. However, computer vision science aims to build machines that can see. It requires the model to be able to extract more valuable information from images and videos than recognition. Generally, it is much more challenging to apply these deep learning models from recognition to other problems in computer vision.

This thesis presents end-to-end deep learning architectures for a new computer vision field: watermark retrieval from 3D printed objects. As it is a new area, there is no state-of-the-art on many challenging benchmarks. Hence, we first define the problems and introduce the traditional approach, Local Binary Pattern method, to set our baseline for further study. Our neural networks seem useful but straightforward, which outperform traditional approaches. What is more, these networks have good generalization. However, because our research field is new, the problems we face are not only various unpredictable parameters but also limited and low-quality training data.

To address this, we make two observations: (i) we do not need to learn everything from scratch, we know a lot about the image segmentation area, and (ii) we cannot know everything from data, our models should be aware what key features they

should learn. This thesis explores these ideas and even explore more. We show how to use end-to-end deep learning models to learn to retrieve watermark bumps and tackle covariates from a few training images data. Secondly, we introduce ideas from synthetic image data and domain randomization to augment training data and understand various covariates that may affect retrieve real-world 3D watermark bumps. We also show how the illumination in synthetic images data to effect and even improve retrieval accuracy for real-world recognization applications.

May 17, 2021

Acknowledgements

First and foremost, I would like to take this opportunity to express my deepest gratitude and respect to my supervisor Associate Prof. Ioannis Ivrissimtzis, who consistently gave me support during my PhD time at the Durham University. I benefited greatly from his encouragement and constructive advice. I feel lucky to work with him and look forward to maintaining the collaboration in the future.

My parents, Mr Zhongbo Zhang, Mrs Hesong Chen also deserve my cordial gratitude. Their love, support and encouragement have always been the source of my strength and the reason I have progressed so far. I also appreciate the support from my best friend in the last five years, Dr Ning Jia, who is there for me in my time of need.

For my PhD research, I would like to express my sincere thanks to Prof. Holly Rushmeier (Computer Graphics Group, Yale University, USA) for inviting me for academic visiting and Associate Prof. Herbert Gangl (Pure Mathematics: Algebra & Number Theory, Durham University, UK) for instructing me 3D printing techniques. I benefited a lot from the valuable research experience from them. I would also like to thank the colleagues at Durham. I get motivated a lot from them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

By the term watermarking, in this thesis, we mean any information embedding on a physical object or a digital file. Watermarks can have various properties, that is, can be visible or invisible, robust or fragile, and they can serve various purposes, such as an object or file authentication, copy control, protection against unauthorised alteration, or dissemination of machine-readable information.

For the intellectual property protection and the information encryption on 3D printed objects as our target applications, this thesis develops algorithms and techniques to support a 3D surface watermark similar in design to DotCode [10, 13].

For common visible digital watermarking, such as bar code, it is printed either on plain object surfaces or presented on plain digital surfaces, with clear black-and-white edges. The clear and smooth background and sharp colour contrast front code result bar code to be easily readable for any devices. However, our designing 3D surface watermark bumps own the same colour as 3D printed objects, hence it is hard to distinguish these bumps from the background. What's more, because these bumps are stereoscopic, their 2D imaging in a photo is severely affected by around illumination environment. Therefore, there are no fix features to extract. Moreover, the process from 3D to 2D, it exists a complicated geometry transform. Besides, there are many random noises in 3D printing progress. Thus, compared with retrieving 2D watermarking, we have to face more and harder retrieval challenges in computer vision.

In the past few years, deep Neural Networks (DNNs) have transformed the field

of computer vision, extending its domain of application into many different new directions. They have been firmly established as the state-of-the-art in various fundamental computer vision tasks, such as segmentation, object recognition, or scene classification. Nevertheless, their suitability for various specific applications still awaits to be verified, especially in situations where there is a shortage of annotated data. Thus, DNNs was a natural choice of technique to be applied to the problem of watermark retrieval from physical objects, and we believe that the research presented in this thesis justified the suitability of our choice.

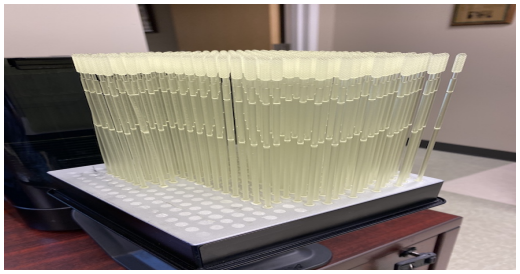
1.1 Additive Manufacturing

Additive Manufacturing (AM) is a well-known name describing the technologies where a computer controls the process of building 3D objects by adding layer-upon-layer of material, whether the material is plastic, metal, concrete [1]. The early use of AM in Rapid Prototyping focused on the production of models and prototypes, especially for short series productions. There, tooling for casting or injection moulding would have been a too laborious and slow process. Since the adding of layer-upon-upon approach allows the AM process to produce almost any designed geometry, there are also many applications of AM technologies in manufacturing end-use objects that are otherwise very expensive or even impossible to make with traditional manufacturing methods. In particular, more recently, AM has been used to fabricate end-use products in medical, automotive, aerospace and even fashion. [2]

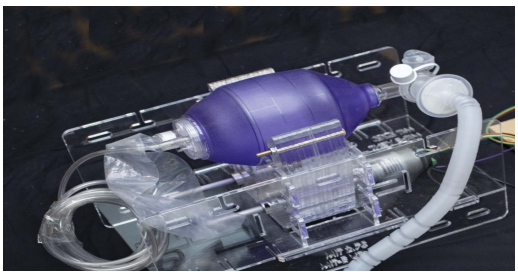
In a rather contemporary note, the emergency of the Covid-19 pandemic is bringing 3D printing technology to the public attention as a tool for tackling severe shortages of personal protective equipment, from face shields to hospital ventilators. When the demand temporarily exceeds conventional production capacity, the versatility of 3D printing technology means that it can be used to produce life-saving emergency equipment.

Thus, a research team from the Czech Institute of Informatics uploaded a design that allows anyone owning a 3D printer to make their mask [7]. A company in Ohio developed a machine which can print out 100,000 nasal swabs [3]. Stratasys has

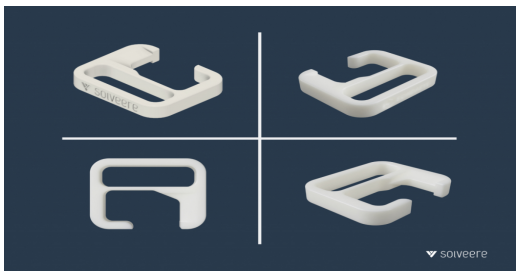
(a) CIIRC RP95 Mask. [7]



(b) 3D-printed nasal swabs. [3]



(c) The printed ventilators. [5]



(d) The hand-free opener. [6]

Figure 1.1: Various 3D printing applications

launched the Covid-19 response website and launched an innovative challenge with Massachusetts General Hospital to design printable ventilators [4]. Engineers in the U.S. First Marine Corps Wing converted their work area, which is commonly used to 3D print parts for aviation maintenance, to a personal protective equipment factory to supply the U.S. Naval Hospital [8]. At a personal level, desktop enthusiasts use 3D printing to produce objects such as hands-free door openers [6].

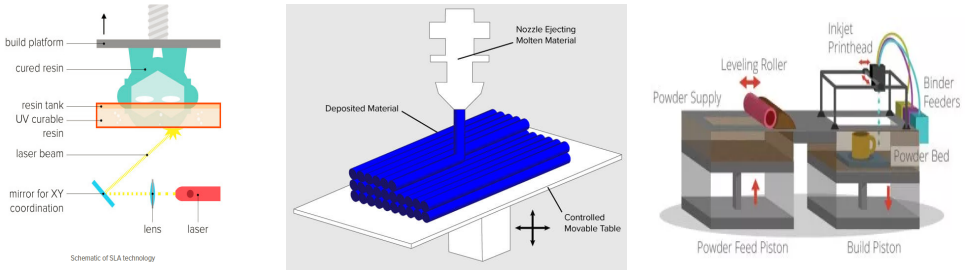
1.1.1 Business models and legal issues

For a long time, 3D printing was considered only suitable for prototyping and one-time printing. There was a misconception that 3D printing is too expensive and that 3D printed parts do not have the desirable attributes found in products manufactured with traditional methods. These misunderstanding has led to a perception that AM is not suitable for use as a production technology. However, 3D printing has now reached the potential to surpass traditional manufacturing methods in small and medium-size batch production. However, the adoption of any new

technology does not happen automatically. When businesses have to evaluate all the risks and costs of transitioning from traditional manufacturing methods, they must study how the entire supply chain will be affected, how quality control change, and how employees will master the new system. A report from Acumen Research and Consulting (ARC) claims that we are at a tipping point, where a substantial number of large companies have already adopted AM in some way, and their latest report predicts that the global 3D printing market will reach about USD 41 billion by 2026 [14].

As 3D printing is disrupting decades worth of legacies in some manufacturing industries, there are many legal issues of concern, including patents, design rights, copyrights and trademarks [15]. In many cases, there are no legal precedents to inform how existing laws should apply if, for example, individuals or amateur communities begin to manufacture items for personal use or non-profit distribution. Moreover, as more and more 3D printers begin to enter people's homes, and traditional relationship between the house and workplace is eroding, it is becoming increasingly easy for amateurs to transfer and exchange designs for new objects worldwide. As a result, many online social platforms have evolved to support the file exchanging 3D printing community [15]. Dedicated sites, such as Pinshape [187], Thingiverse [188], and MyMiniFactory [189], have been created to allow users to post 3D files for anyone to print, thereby reducing the transaction costs of sharing 3D files.

These increasingly significant social interactions between users, and the lower cost of 3D printers, not only bring more intellectual property (IP) concerns but also magnify problems related to the manufacturing of controlled items, such as guns. A few years ago, the US government issued a memorandum stating that *availability of free digital 3D printer files for firearms components, and difficulty regulating file sharing may present public safety risks from unqualified gun seekers who obtain or manufacture 3D printed guns*, and that *proposed legislation prohibiting 3D printed weapons may deter but not wholly prevent weapon production. Even if the new regulations ban this practice, these 3D printable documents can still be illegally distributed online.* [16]. Therefore, for the further sustainable development and



(a) SLA (b) FDM (c) 3DP

Figure 1.2: Three types AM technologies applications.

adoption of 3D printing technologies, the issue of how to protect the IP and how to guarantee safe use and tractability will play an important role.

1.1.2 Review of 3D printing technologies

AM technology can be broadly divided into three types (Figure: 1.2). Stereolithography (SL), which was introduced in 1987, is a process by which a laser solidifies thin layers of ultraviolet (UV) light-sensitive liquid polymer. The leading commercial technologies based on stereolithography are SeteroLithography Apparatus (SLA)(Figure: 1.2a), Solid Object Ultraviolet Plotter (SOUP), and Electro-Optical Systems (EOS). Later in the development of Additive Manufacturing, non-SL systems were introduced, which were based on processes extruding thermoplastic materials in filament form to produce objects layer by layer. The fused deposition modelling (FDM)(Figure: 1.2b) is one such technology which has found many commercial applications. Finally, low-cost 3D printers were introduced, based on a technology that utilised an inkjet printing mechanism to deposit wax material or apply a small amount of binder on the powder of either starch or plaster based material, to form a layer of the object. MIT's inject printing (3DP) (Figure: 1.2c) is an example of a successful commercial product based on that technology. Currently, nearly all new generation machines are based on one above three technologies [9].

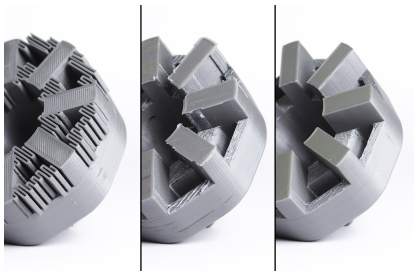
According to the 2020 3D printer comparison guide [17], since FDM printing patents expired in 2009, many companies had opportunities to develop their own devices based on variants of that technology, which resulted in prices for FDM printers dropping from $10,000 to less than $1,000 [18]. With the cost of FDM 3D

printers continuing to drop, the market for FDM 3D printers grows, and naturally, the FDM filament marked followed suit. The most widely used 3D printer filament types are ABS (acrylonitrile butadiene styrene) and PLA (polylactic acid). Of course, there are also many other commonly used materials, including wood-filled, flexible and even conductive materials. In 2012, Filabot, as a plastic company built a 3D plastic extrusion system, which could convert a broader range of plastics into filament for use in 3D printers [19]. The Filabot devices are not only environment-friendly but also can help the user of FDM 3D printers to become self-sufficient. We not here, that as one would have expected, these days there are a lot of 3D printing videos on Youtube, with some YouTubers being very enthusiastic about their research and presenting in great detail. Among such videos, it is clear that FDM technology is the easiest to learn and use. Finally, besides the advantages as mentioned above, such as affordable price, ease of use and powerful functions, the FDM 3D printers have been the easier to scale down to desktop 3D printing size. All, these factors together make FDM the most popular consumer-grade desktop 3D printer technology, in what is arguably a quite competitive market.

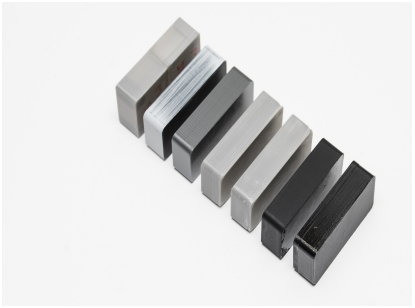
Most of the 3D printed objects used in the experiments in this thesis were printed in an original Prusa i3 MK2/S Multi Material 1.0, which is an updated version of original Prusa i3 MK2. It allows users to print not only in a single filament but use up to 4 colours or materials simultaneously. However, even the best FDM 3D printer can not avoid the need for post-processing, which arguably is the least attractive aspect of 3D printing, but essential for making end-use products.

Any 3D printing technology such as FDM that needs support parts, support removal is the first post-processing step. The standard support structures are designed in a way that usually makes them easy to remove from the printed object. Typically, users will use toothpicks and needle-nose pliers to clean the support material in hard-to-reach places such as holes or hollows [20]. We note that support removal does not interfere with the overall geometry of the real object, that is, this first post-processing step does not remove artefacts in the form of layer lines, striations, or blemishes on the printed surface.

Thus, after removing the support structure, the remaining artefacts still im-



(a) Original print with support attached, poor support removal and good support removal (left to right).

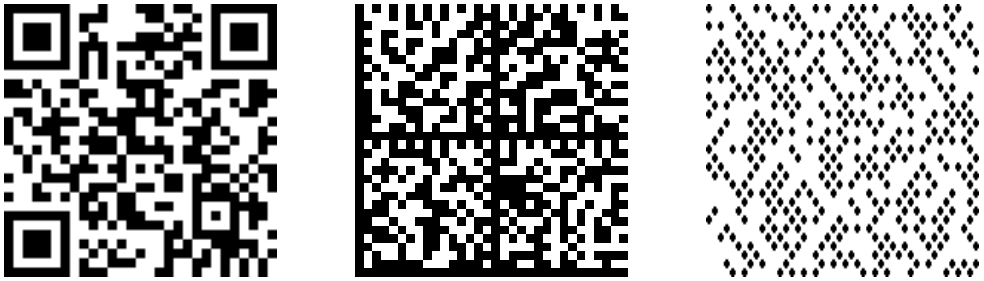


(b) Post processed FDM prints (from left to right): Cold welding, gap filling, unprocessed, sanded, polished, painted and epoxy coated.

Figure 1.3: Left: support removal post-processing. Right: other post-processing methods.

pact negatively on printing accuracy and appearance (Figure: 1.3a). For example, as artefacts in the form of layer lines often appear on FDM prints, when smooth surfaces are required, further post-processing becomes unavoidable. The most common methods for further FDM post-processing include sanding, cold welding, gap filling, polishing, priming and painting, vapour smoothing, dipping and epoxy coating [20](Figure 1.3b). Unfortunately, due to limitations of our lab equipment, our in-house printed objects were only post-processed for support removal followed by simple sanding. Sanding can produce a smooth surface finish. However, it can be challenging to apply to intricate surfaces with many small details. Moreover, if sanding is done too aggressively, a lot of material will be removed, impacting the overall accuracy of the print.

In some of our experiments, we accepted that the rough surface finish of our 3D printed objects would add experimental difficulty and reduce the accuracy of the watermark retrieval algorithms. In other experiments, we opted for professionally printed objects by an external provider of the 3D printing services.



(a) QR Code (b) Data Matrix Code (c) DotCode

Figure 1.4: The three most popular 2D barcodes encoding, in this example, same information.

1.2 Watermarking

1.2.1 Barcodes

As technology advances and the needs of businesses or customers change, finding the right information reading technology can be a challenge. Barcodes are the ubiquitous technology for embedding machine-readable information on a physical object. Beyond the widely used 1D barcodes, moving from 1D scanning to 2D scanning, 2D barcodes have recently gained popularity, replacing rapidly 1D barcodes in all but the most trivial applications.

2D barcodes are readable from any direction, on any flat surface. Due to the redundancy offered by their higher information capacity, they contain a higher Error Correction Level (ECL), keeping them roust during their whole lifecycle. That is, compared to 1D barcodes, there is a higher percentage of the 2D barcodes that can be damaged before it is no longer readable, and hence, higher ECL can be used to account for some damage to the physical object carrying the code.

Currently, the most popular 2D barcodes are QR Codes [11], Data Matrix Codes [12] and DotCodes [10, 13] (Figure: 1.4). More details about 2D barcodes will be discussed in Section 2.1.

In this thesis, the 3D printed watermarking is based on the DotCode design, which as a machine-readable code that can be read without requiring the fixing of the orientation of the object and can be printed on non-flat surfaces, such as cylindrical, round, or oval surfaces. We note that while in this thesis, we print almost exclusively

on flat surfaces, extensibility to non-flat surfaces was an important consideration, and has been tested in small proof-of-concept experiments.

1.3 Watermarking

The primary purpose of watermarking is to make it difficult for the original copy to be appropriated, or imitated fraudulently. The first modern watermark, which was embedded on paper, can be traced back to Fabriano in Italy, in 1282 [128]. Over the years, the types of watermark have evolved from the earliest watermarks that typically varied the thickness of the paper, to the embedding of shadows in the paper, to the watermarking for digital media. Digital watermarking is an extension of the original watermarking concept to the digital world, and it is typically used to verify the integrity of the content or to recognise the digital content's owner and facilitate copyright protection. However, beyond these applications, digital watermarking is also used in many real scenarios for purposes such as source tracking, broadcast tracking, and hidden communication.

A simple online search shows that images with watermarks on Facebook, Twitter, Google+, and on the internet, in general, are becoming more and more common. Even more complex watermarks are used on banknotes, postage stamps and various government and commercial documents.

Given the range of potential applications of digital watermarking, the interest in developing attacks on it is also high, as well as, in response, methods for preventing intentional or unintentional damage to the watermarks. Here, we are interested in such attacks mainly as a means for testing the robustness of the watermarking algorithms we develop. Below is a standard categorisation of attacks, given in [135]:

- **Image processing attacks:** several image processing attacks can be applied to test the robustness of the watermarking scheme. Such attacks include filtering, perceptual remodulation, JPEG compression distortion.

- **Geometric attacks:** these attacks do not remove the digital watermark itself, but disrupt the synchronisation of the watermark detector with the embedded data. Such attacks include image scaling, rotation, image clipping, lin-

ear transformations, bending, wrapping, perspective projection, collage, templates.

- **Cryptographic attacks:** the target of these attacks is possible encryption of watermark, which might have been added at the embedding stage as an extra layer of security, and it is attacked for either reading the encrypted watermark, or for ensuring that it has been removed. Such attacks include the brute-force attack, Man-in-the-Middle (MitM) attack, replay attack, side-channel attack, power analysis attack, and timing attack.

- **Protocol Attacks:** these attacks target the entire pipeline of the watermarking application rather than the embedded information itself. There are two main types of protocol attacks, the invertibility attack and the copy attack.

- **Other sophisticated attacks on block-based fragile watermarking:** these attacks target the localisation and recovery properties of such watermarking schemes. One example is the Vector Quantization (VQ) attack, which builds a pseudo image from a VQ codebook generated from a set of watermarked images.

1.3.1 3D digital watermarking

The goals and principles of 2D image and 3D model verification have many similarities. However, the methods used in the image domain do not usually extend directly to 3D objects. Compared to the relatively mature theory and application domains of 2D digital watermarking, the watermarking methods for 3D models are still somewhat immature. It is mainly due to the complications encountered when dealing with arbitrary topologies, the irregular sampling of 3D meshes, and the complexity of possible attacks on watermarked meshes.

A mesh may contain various types of information, including geometry, colour on vertices, directions of vertex normals, and images used for texture mapping. A type of attributes may well be modified, or even completely removed, without significantly changing the end-use of the model in specific application scenarios, that is, the perceived quality of the graphics generated from the mesh will not deteriorate,

or it might even not change at all. Another complication arises from the fact that is some critical computer-aid design applications the watermarked meshes are not only mainly evaluated by the somewhat subjective human visual system but also by some strict objective indicators as, for example, performance in sophisticated simulations.

Similarly to image watermarking, the evaluation of watermarking methods for 3D models requires the analysis of the trade-off between capacity, robustness, and imperceptibility. These properties are often contradictory. For example, increased watermark strength often leads to increased robustness, but often reduces the visual quality of the watermarked meshes and runs the risk of making the watermark perceptible. Similarly, information redundancy can significantly improve robustness, but at the same time, inevitably reduces capacity. We note that local adaptive geometric analysis seems in many cases to be a handy tool into finding optimal watermark parameters [28].

1.3.2 Watermarking for 3D printing

Digital 3D watermarking is used extensively in 3D printing-related applications. Indeed, the protection of intellectual property afforded by digital watermarking is deemed necessary by companies, independent designers, or amateurs who upload online their proprietary 3D files for trade or just for sharing, taking the risk that these files can be maliciously altered, or re-sold. What content owners are currently doing is either to embed a hidden 3D digital watermark in their 3D files and then to track the digital watermark online, or directly print exclusive logos on their printed objects.

For example, the 3D printing service provider Treatstock [21] is using *Watermark 3D* [22] to allow designers who upload files to embed custom information. There is also optional password protection that controls the rights to retrieve the watermark. Once the original file is uploaded, a download link for the watermarked file will be sent via the email. Watermark 3D claims that the embedded watermark is "almost impossible" to be detected, copy or alter. Meanwhile, the embedded information will not"change the structure" nor leave any"visual mark" on the final design. However,

there are still limitations, the most fundamental of which is that the watermark protects the digital file, but not the physical printed object itself. Indeed, since there is not any physical distortion of the 3D model caused by the watermark, after sending the STL file with the embedded watermark to a 3D printer, the embedded IP protecting information will not be present on the printed object.

1.3.3 Watermark retrieval from 3D printed objects

3D X-ray is high-end and also high-cost, technology offering as a non-destructive method to retrieve internal and external geometric features of 3D printed objects. It is a natural choice of technology for retrieving watermarks embedded deep inside the 3D printed part.

Scientists at the Department's of Energy *SLAC* National Accelerator Laboratory are using X-rays to observe and understand how the additive manufacturing process for printed metal parts leaves defects and how to prevent them. These studies aim at helping manufacturers produce spare parts on-site, whether that is in a factory, on a ship, an aeroplane, or even in remote areas in space, increasing the efficiency of the maintenance processes by, ideally, not having to store thousands of spare components [23].

The quality of CT data depends on the energy level of the X-ray beam, the signal-to-noise ratio of the detector, and the stability of the apparatus. The Scientists in SLAC, using high energy X-ray beams can create micron-resolution images showing what happens as metal layers build up. They also detect reflected X-rays to analyse the atomic structure of the material as it changes from solid to liquid during melting and then is cooling again to return to solid state [23]. Analysing these data, they try to model the 3D printing processes with more complex geometric forms and infer which type of 3D printing technology is most suitable for manufacturing spare parts.

At the lower end of application scenarios, most consumers can only afford commodity cameras to acquire images, while most of the companies may only be able to afford specialised equipment well below the level of a heavy X-ray synchrotron. Ideally, in these low-level applications, people should be able to infer the meaning of a 3D watermark using simple and portable devices, and preferably, the now-ubiquitous

smartphone cameras. Thence, we decided to design a kind of 3D watermark that can be retrieved by commodity smartphone cameras.

1.3.4 2D barcodes

2D barcodes readable by 2D machine vision systems have been in use for many years, proving their value in a variety of application scenarios. The 3D watermarks used in this thesis, encode information is a way that is very similar to 2D barcodes, essentially in the form of a 2D bit array, the main difference being that instead of the 2D image intensity or colour values, the information carrier here is the fine surface geometry in the form of surface bumps arranged in a grid. We note however that, despite this fundamental difference, the optimisation of the retrieval algorithms for 2D image and 3D geometry barcodes share the same considerations: environmental lighting, type and quality of optical lenses, digital image capture sensors, image processing software tools and communication interfaces.

Assuming a simple 2D machine vision system, that is, without the use of 3D scanners or other more specialised equipment, the information embedded on the 3D geometry of the surface has to be retrieved from a 2D image, which does not explicitly contain height information. It entails various challenges, most of which are also encountered in the retrieval of 2D image barcodes. First, and most importantly, what the camera captures is the projection of the physical object from a specific camera viewpoint, and different viewpoints may correspond to significantly different projections. Typically, if the 2D or the 3D barcode is embedded on a perfectly flat surface, if the camera view is perpendicular to that planar surface, and if a suitable focal length means that the barcode is in sharp focus, then we should be able to retrieve the embedded information with high accuracy. Variations in the lighting environment is another issue which can adversely affect retrieval accuracy. For example, too much light or unwanted shadows can adversely affect the sharpness of edges of a 2D image barcode or the sharpness and position of the edges corresponding to the surface features encoding information in a 3D barcode.

Over the years, various formats of 2D barcodes have been developed. We have, for example, the prevalent *QR code*, the *Aztec code*, or the *Matrix codes*. They

most often appear in the form of black squares arranged in white background, but each one uses its encoding system with its trade-off between capacity, robustness, and decoding efficiency. We note that while these codes may exhibit some retrieval resiliency when embedded on non-planar surfaces, their design assumes embedding on planar surfaces. Plain 2D bit arrays will represent the information we embed on 3D printed objects with our watermarks. That is, here, we neither propose any novel 2D code scheme nor use any of the existing ones. We believe that finding an optimal for our purposes 2D code is a a fascinating problem. However, it seems orthogonal to the fundamental computer vision problem we focus on.

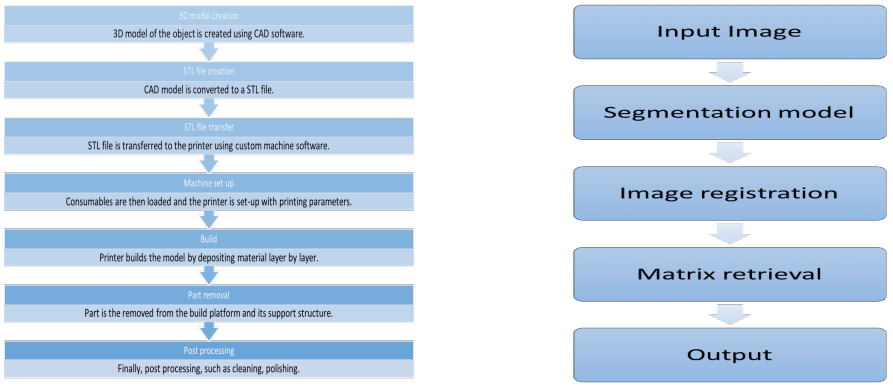
1.4 Deep neural network and synthetic image datasets

Semantic segmentation, that is, the ability to segment unknown images into their different parts and objects is widely recognised as one of the biggest challenges in computer vision. Generally, segmentation is considered a more fundamental problem than object detection, because segmentation does not require recognition.

Semantic segmentation without knowing the exact identity of all objects in the scene is an integral part of our visual understanding process, which can provide us with powerful models to understand the world and can also be used to improve or enhance existing computer vision techniques. In particular, humans are capable of performing image segmentation without knowing what the object is. For example, satellite image, or medical X-ray scans usually contain multiple unknown objects, which nevertheless, usually, can be segmented by experts as the first step towards scene analysis and understanding.

Traditional image segmentation algorithms are usually based on clustering, and usually make use of other information from the contours and edges of the image [24]. For example, in the simplest case, satellite image segmentation can be successfully performed by clustering pixels based on an estimation of the wavelength. It means that clustering can be done by similar grouping similar pixels which are also located near in the space.

There are numerous successful clustering methods which predate deep learn-



(a) 3D watermarking design and print flow (b) Printed 3D watermarking retrieval process

Figure 1.5: All the blocks of my methodology

ing. One of the most notable and widely used is based on Markov processes for modelling [25]. Another more recent method is to combine contour detection with layered methods [26, 27]. However, in this thesis, only the current state-of-the-art methods will be introduced, because recent advances in deep learning have rendered many older methods obsolete, by achieving much better performance on widely used benchmark datasets.

While deep learning methods have brought tremendous progress in object detection and segmentation, their use in any specific application scenario is not without issues. In particular, the utilisation of their full power often requires a quantity and quality of data. A common method to circumvent this problem is to synthesise training data. In particular, physics-based rendering software can be used to obtain realistic images by accurately simulating a real-world optical flow according to the various physical laws. Moreover, techniques such as style transfer learning and random domain adaptation can be employed as low-cost methods for reducing the gap between real data and synthetic data.

In this thesis, we use physics-based renders to generate synthetic data to remedy the problem of having a limited amount of real-world training data at our disposal. We then proceed to study further the effect of the various parameters of the process of synthetic image data generation to the performance of the deep neural networks. Figure 1.5 shows all the blocks of my methodology.

1.5 Contribution and Thesis Outline

The overarching objective in this thesis is to design camera view independent 3D watermarks for 3D objects that are robust under a considerable amount of manufacturing noise, for example, retrievable when printed with the FDM technique, and develop the corresponding retrieval algorithms. By view independent watermark, we mean that when the watermark can be photographed from any direction by a general smartphone camera and the extracted information will always be the same.

While the proposed 3D watermarks can be printed on non-flat surfaces, here we test almost exclusively the developed techniques on flat surfaces. The 3D watermarks I design are embedded on flat surfaces of 3D printed objects and encode information in 20×20 bit arrays, where a bit value equal to 1 corresponds to a semi-spherical bump on the object's surface.And each printed watermarking bump size is about 1mm. We study the effect of several variables, such as the surface roughness, texture and colour of the filaments, lighting conditions in the environment in which the watermark is retrieved, and propose extraction algorithms ranging from traditional methods to, mostly, deep neural networks. To address the problems caused by the multitude of variables, we developed several extensions of classic DNN architectures and conducted a systematic study of the effect of the illumination model in the generation of synthetic image training data.

1.5.1 Contributions

The contributions of this thesis are listed as fellows.

An initial qualitative study of the various possible approaches to the problem of watermarking 3D printed objects

We propose a taxonomy of 3D watermarking techniques by considering their most critical characteristics, such as watermark visibility, view dependence, and the shape of the carrier surface or volume. For example, regarding watermark visibility, we separate two classes of techniques. The first class consists of techniques that embed visible 3D watermarks on the surface of the object, or below the surface of transparent parts of the 3D printed objects. The second class consists of techniques

embedding invisible 3D watermarks hidden under the surface of non-transparent parts of the object. This initial study covers a gap in the literature regarding 3D watermarking studies tailored to real 3D printing application scenarios.

We also present quick proof-of-concept solutions for various of the classes of techniques in the taxonomy. See Chapter 3 for the taxonomy and some techniques we did not pursue to great length. In the main body of the thesis, Chapters 4 to 7, we go deeper on visible watermarks on flat surfaces.

Single Image Watermark Retrieval from 3D Printed Surfaces via Convolutional Neural Networks

In chapter 4, we propose and analyse a method for watermarking 3D printed objects, concentrating on the watermark retrieval problem. The method embeds the watermark in a planar region of the 3D printed object in the form of small semi-spherical or cubic bumps arranged at the nodes of a regular grid. The watermark is extracted from a single image of the watermarked planar region through a Convolutional Neural Network (CNN). Experiments with 3D printed objects, produced by filaments of various colours, show that in most cases the retrieval method has a high accuracy rate.

In chapter 5, we further improve the method using a more advanced network architecture. Extensive experimentation with images captured from various camera views, under various illumination conditions and from objects printed with various material colours, shows that the proposed method generalises well and achieves the level of accuracy required in practical applications.

Watermark retrieval from 3D Printed Objects via Synthetic Data Training

We present a deep neural network-based method for the retrieval of watermarks from images of 3D printed objects. To deal with the variability of all possible 3D printing and image acquisition settings, we train the network with synthetic data. The main simulator parameters such as texture, illumination and camera position are dynamically randomised in non-realistic ways, forcing the neural network to learn

the intrinsic features of the 3D printed watermarks. At the end of the pipeline, the watermark, in the form of a two-dimensional bit array, is retrieved through a series of simple image processing and statistical operations applied on the confidence map generated by the neural network. The results, see Chapter 6, demonstrate that the inclusion of synthetic Domain Randomisation data in the training set increases the generalisation power of the network, which performs better on images from previously unseen 3D printed objects. We conclude that in our application domain of information retrieval from 3D printed objects, where access to the exact computer aided design (CAD) files of the printed objects can be assumed, one can use inexpensive synthetic data to enhance neural network training, reducing the need for the labour intensive process of creating large amounts of hand labelled real data or the need to generate photorealistic synthetic data.

Illumination Models for the Generation of Synthetic DNN training Datasets

The use of computer-generated images to train Deep Neural Networks is a viable alternative to real images when the latter are scarce or expensive. In Chapter 7, 3D printing becomes the means rather than the goal of our study, and we use it to study how the illumination model used by the rendering software affects the quality of the generated images. We created eight training sets, each one with a different illumination model, and tested them on three different network architectures, ResNet, U-Net and a combined architecture developed by us. The test set consisted of photos of 3D printed objects produced from the same CAD models used to generate the training set. The effect of the other parameters of the rendering process, such as textures and camera position, was randomised.

Our results show that the effect of the illumination model is important, comparable in significance to the network architecture. We also show that both light probes capturing natural environmental light, and modelled lighting environments, can give good results. In the case of light probes, we identified as two significant factors affecting performance the similarity between the light probe and the test environment, as well as the light probe's resolution. Regarding modelled lighting environment, a

similarity with the test environment was again identified as a significant factor.

1.5.2 Thesis Outline

The rest of this thesis is organised as follows. In Chapter 2, firstly, we review the relevant literature of object detection methods, including traditional algorithms and deep neural networks methods. Then we review state of the art about 3D watermarking and synthetic rendering data. We also introduce some fundamental knowledge on domain adaptation, light probes and the measures of a test's accuracy. In Chapter 3, we introduce a taxonomy of 3D watermarking techniques for 3D printed objects and some quick proof-of-concept techniques from various parts of the taxonomy. In Chapter 4, we replace the proof-of-concept traditional image processing method for watermark retrieval from flat surfaces with a convolutional neural network algorithm which can retrieve 3D watermarking information with satisfactory accuracy from a single 3D printed objects. In Chapter 5, we build the work in Chapter 4, improve the network architecture and add more experiments, exploring how different training methods affect the final retrieval precision. In Chapter 6, we incorporate synthetic images data into our training datasets to retrieve 3D watermarking from real images of 3D printed objects. In Chapter 7, we use various synthetic image generation methods and a new DNN framework to explore how the illumination model of the synthetic image data affect the accuracy ratio in the retrieving of watermarks from real objects. Chapter 8 concludes this thesis and suggests a direction for further research.

1.6 List of Publications

The full list of publications proeduced during my PhD project on 3D watermarking on 3D printed objects is as follows:

- **X.Zhang**, N.Jia, and I.Ivrissimtzis."A study of the effect of the illumination model on the generation of synthetic training datasets." *to be submitted to Computers & Graphics.*

- **X.Zhang**, N.Jia, and I.Ivrissimtzis. "Watermark retrieval from 3D printed objects via synthetic data training." In *the 28th International Joint Conference on Artificial Intelligence Workshops*, 2019.

- **X.Zhang**, Q.Wang, T.Breckon and I.Ivrissimtzis. "Watermark Retrieval from 3D Printed Objects via Convolutional Neural Networks." *arXiv preprint arXiv:1811.07640*, 2018.

- **X.Zhang**, Q.Wang, and I.Ivrissimtzis. "Single image watermark retrieval from 3D printed surfaces via convolutional neural networks." *Proceedings of the Conference on Computer Graphics and Visual Computing.* Eurographics Association, 2018.

Chapter 2

Literature Review

This chapter reviews work in the field of watermarking and computer vision. Particular attention is paid to image segmentation , data augmentation and image processing algorithms. This is followed by a description about 3D model design that I used in my experiments. Finally, a subset of performance metrics will be discussed.

2.1 Watermarking

2.1.1 2D Barcode

As mentioned in Chapter 1, more details about the QR Code, Data Matrix Code and DotCode will be introduced in this section.

QR Code

QR refers to "Quick Response" (Figure:1.4a), and QR Code has been widely used by anyone with a smartphone, playing the role of bridging the real-world and the digital universe. A QR Code can store up to maximally 2953 bytes, which equal to 7089 numbers or 4296 alphanumeric characters, or even 2953 Kanji characters [121]. The three distinctive square structures at its corners are QR code's finder pattern, which is used by scanners to recognise and read the code [114]. To improve the reliability of QR code under special scenarios, such as being partially contaminated, damaged or occluded, the Reed-Solomon Error Correction algorithm [119] is intro-

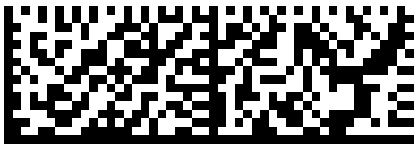
duced, whose working mechanism can be summarised as complement missing data by calculating a preset polynomial. Further, there are four error correction levels (ECLs) based on the amount of data to be recovered, ranged from 7% up to 30%. A higher level indicates a lower storage capacity, since recovering a certain amount of data requires twice the amount of Reed-Solomon Code [119]. The superiority of QR code includes high information encoding capacity per unit area, orientation invariant, high tolerant to information loss, *etc.* Thus one of its main utility is redirecting URL to establish "virtual stores", to store bank account and credit card information for making online transactions.

Data Matrix code

The Data Matrix code (Figure: 1.4b) is also a 2D code typically in a square shape, which encodes information with a number of black and white "cells". As shown in Figure: 1.4b, it has an "L" shaped "finder pattern" composed of two solid adjacent borders (left and bottom) that helps identify the code. A matrix code symbol can store up to 1556 bytes, equal to 3116 numbers or 2335 alphanumeric characters from the full ASCII range [120]. Comparing with QR code, Data Matrix code's ECL can be automatically determined up to 30% by the code size and the remaining storage capacity. The Data Matrix codes are trusted to be more secure (less hackable) and are preferred where high security is the top priority [119]. While both Data Matrix codes and QR codes are scalable [114], Data Matrix code is more compact than the QR code, that it is applicable within a minimum area of 0.1mm×0.1mm [120]. Thus it is preferred in special application scenario such as medicine/surgical instruments that have minimal space. In addition, to fit the unique requirement of limited height or high-speed printing, Data Matrix codes can be in rectangular form, with a maximum capacity of storing 98 numbers. A rectangular symbol and a square symbol encoded with the same information are shown in the Figure below (Figure: 2.1).

DotCode

The latest updated of DotCode (Figure 1.4c), designed by Dr Andrew Longacre and published in 2009 [122], already demonstrated great success in the European tobacco industry (Figure 2.2a). It does not require the precise length of lines or



(a) Rectangle

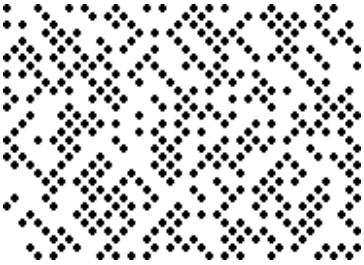
(b) Square

Figure 2.1: A rectangular symbol versus a square symbol.

spacing between dots, thus is robust against high conveyor speeds up to 1000 codes per minute and vagaries of ink technology [123], while other 2D barcodes would suffer from information loss. Plus its shape is flexible in the form of a square or more ribbon-like without max bytes limitation like rectangle matrix code (Figure: 2.2b). Also, the DotCode is machine-readable in all positions, thus desirable for use in rough conditions. For example, the products out of metal casting require heating in a fire or furnace, and it is nearly impossible for other types of codes to be printed accurately onto the surface, plus for items like a metal shaft with the minimal height it is difficult to contain barcodes with special finder patterns. In contrast, DotCode that can be printed in flexible shape with either concave or bump dots is suitable for such a scenario. With its unique discrete dot design, which no other dots (neither neighbouring matrix nor alien) are allowed within a zone of twice the dot diameter around the total DotCode matrix, and which no additional dots other than belonging to the matrix are allowed within the DotCode matrix [124], even a relatively poor-quality printed DotCode with lightly crumpled or crookedly dots can still maintain robust "scannability" [123].

For real world applications, e.g. medical field has intense demand for embedded tags on the medicine containers of various shapes (cylindrical/round, square, rectangular, oval, *etc.*), to encrypt prescription and expiry date. The current mainstream 2D barcodes, such as the QR Code and Data Matrix Code above-mentioned, are able to meet those demands. However, they can not satisfy the commercial level recognition rate when they are used on a non-planar surface. On the other hand, though the Radio frequency identification (RFID) can embed information on tags

(a) DotCode on a pack of tobacco product for traceability. [125]



(b) The up is ribbon-like; The bottom is square [123]

Figure 2.2: Left:DotCode application.; Right: Two representative DotCode symbols.

adhered on curve surface [126], the cost of RFID tag is 100 times higher than the ordinary 2D barcode [127]. Plus, the information tags embedded with RFID can only be recognised by special RFID reader through radio frequency transmission, which is inconvenient for patients. Shim and Kim designed a robust circular dot pattern code (CDPC) for tagging information on cylindrical or round surfaces [127]. Their experiments proved that the dot-based CDPC outperformed 2D barcodes, such as QR code, in recognition rate on cylinder objects. Based on the above argument, in this paper, we also use a kind of dot-based code for embedding information on 3D printing objects identification.

2.1.2 Digital watermarking

The classes of digital watermarking

As a fairly new subject, digital watermarking has been developed in different directions. We will present an overview of those taxonomies in different ways.

Based on visibility

From human's visual perceptibility, generally, there are two types of digital watermarking:

- Visible digital watermarking: the watermark is embedded in a visible pattern on the object. It can be a logo or a text that marks the ownership of photos and images. On the other hand, for community managers and anyone working with a social network, a visible watermark can as an effective way to promote

brand or website, because when users look at a picture with that watermark, they will know its ownership [129].

- Invisible digital watermarking: The embedded pattern is invisible or, in case of audio content, inaudible. The invisible watermarking has been widely implemented in many fields, including audience monitoring, content integrity, content protection, forensics, user tracing and packaging identification [129].

The significant difference between visible and invisible digital watermarking is that the visible watermarking is an overlay text or logo superposed on top of the picture and can be easily removed, while the invisible watermarking conveys a secret message with subtle modifications by changing pixel values in certain positions. Still, the basic requirement for digit watermarking is the robustness against image editing, such as compression, scaling, colour editing, that the embedded message should always be fully recovered. On the contrary, a visible watermarking is more easily affected by pixel-level manipulations than an invisible one. The simplest way is to crop the visible watermarking or remove it with machine learning/image processing models, e.g. an algorithm released by Google that can erase all visible watermarks from photographs [130]. Hence, photographers introduced a new type of watermarking apart from the aforementioned two: the dual digital watermarking. The dual digital watermarking mixes two classical digital watermarking, where an invisible watermarking is used as the backup for the visible one [129].

Based on domain

Watermarking techniques can be applied in either the spatial domain or frequency domain [131]. Watermarking in the spatial domain has the advantages of low computational cost and easy to implement, via directly changing image pixels to embed watermarks. The most common approach is the least significant bit (LSB) and patchwork, which is done by replacing the least bit of plane, which does not have any visual impact, with the watermark data bits. However, it also brings some serious security problems, because LSB can easily be affected by intentional or unintentional attacks that alter watermark bits. The other category is the frequency domain technique, which is developed later than spatial domain watermarking, includes discrete cosine transforms (DCT) [132], discrete wavelet transforms

(DWT) [133], discrete Fourier transform (DFT) [134], singular value decomposition (SVD) [133] and Karhunen-Loeve transform (KLT) [135].

Based on the extract process Based on the requirement of the original image during the watermarking recovery process, there are three types of watermarking techniques: blind watermarking, non-blind watermarking and semi-blind watermarking. For the blind watermarking, whose notable applications are healthcare, electronic voting system and copyright, it does not require an original image nor the watermark pattern. In the non-blind watermarking system, whose potential application are covert communication and copyright protection, it needs an original and watermarked image to extract watermarking. The semi-blind watermarking system does not need the original image but only the watermark, which is usually used in the fields of image authentication and CAD models.

2.1.3 Printer steganography

Machine identification code (also known as printer steganography) is a digital watermark that some colour laser printers and photocopying machines leave on each printed page. We found that it shares common ground with the 3D watermarking technique proposed in this thesis, but represented in a different form. That encrypted secret digital watermarking ensures that the output of those printers is forensically traceable (Figure: 2.3).

Printer steganography was developed by Xerox in the mid-1980s [136] and is widely adopted by all kinds of printers from different manufacturers, including all the big brands such as Brother Canon, Dell, and Xerox [137].

When printing a document, tiny yellow dots are added to each page and are almost invisible to the naked eye. These yellow dots contain encoded timestamps and serial numbers that link the pages to the specific printer from which they are printed. Those dots appear in yellow because transparent ink or toner absorbs light under the sun and reflects red and green, and eventually the red and green indicator lights turn yellow. For a colour printer with four-colour-printing technique, we are about to see red, green, blue and white in the additive colour primaries system. The four primary colours that make up the colour model are mixed and synthesized to

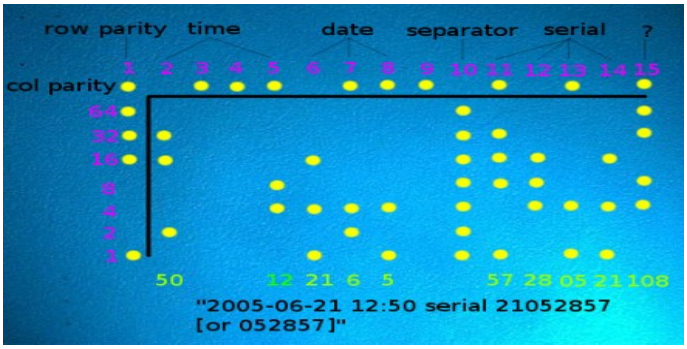


Figure 2.3: Yellow dots found on printed documents under a blue light. [137]

generate many other colours. During colour printing, after a paper is affixed to the photosensitive drum, a positive charge is applied to attack toner. This process is conducted four times with separate scanners and toner cartridges for each of the four colours creating a colour image on the paper. In other words, these yellow dots cover at least three the colour channels at three different layers. The dots are shown on a 2D flat, but they can be seen as a 3D combination in colour channel.

Similarly with machine identification code, 3D watermarking on printed objects also storage much information for forensically tracing and IP protection. Apart from being visible like above yellow dots, it can also be invisible by hiding inside printed 3D objects. Instead of using a select colour, 3D watermarking uses bumps to conceal information.

2.2 Segmentation algorithms

In computer vision, representative applications include image understanding [29,30], object detection [31,32], unsupervised video object segmentation [33–35] and semantic segmentation [36,37], *etc.* Moreover, object detection is an important computer vision task for detecting instances of visual objects in a particular category, such as a person, animal or car, in a digital image. The purpose of object segmentation is to develop a computing model and technology that can provide one of the most basic information required by computer vision applications. Of course, as one of the basic problems of computer vision, object segmentation forms the basis of many other computer vision tasks, such as segmentation, image captioning, and objec-

tion tracking. In recent years the rapid development of deep learning technology has brought new blood to traditional methods for target segmentation, brought a breakthrough, and pushed it to an unprecedented research hotspot. Object segmentation is now widely used in many practical applications such as autonomous driving, robot vision, video surveillance, *etc.*

In this section, we will review the area of object detection in multiple aspects, including traditional segmentation methods, the evolution of critical technologies, such as various deep learning frames, and others extended methodologies.

2.2.1 Challenges and issues

Object detection plays a vital role in image understanding tasks. In computer vision community, the following three characteristics are mainly used to detect objects from an image: spectral features, spatial features (e.g. edge features and texture features), and temporal features (or motion features). In ideal cases such as laboratory conditions where camera angle, illumination intensity and light source location, and the surrounding background of the objects are fixed, thus those three characteristics can easily be extracted and used to separate objects from the background or distinguish between different objects. However, in practice, it is often impossible to maintain the above mentioned ideal condition, where the object features can be very different within the same class, or being very similar among different classes due to the global pixel intensity change (e.g. with poor illumination condition the texture of the objects is indiscernible. Thus two different objects with similar shape but different texture can be classified as the same type). Based on these ideal conditions, features extractions give good results. However, in practice, many other challenging conditions may arise and interfere with this process. The list is as below:

- Noisy image: This is due to the quality of the image source, such as images acquired through a webcam or compressed images;
- Camera automatic adjustments: Many modern smartphone cameras have autofocus, automatic gain control, automatic white balance, and auto-brightness control. These adjustments change the graduation dynamics between different

frames in the sequence;

- Illumination changes: They can be either the progressive change of natural light through a period of time or sudden change in an indoor scenario;
- Bootstrapping: During training, there are no key features in some images, which makes it impossible to calculate a representative object detection image;
- Camouflage: The pixel characteristics of the foreground object can be included in the modelled background. The segmentation model struggles to find the exact silhouette of the object when the foreground and background shares similar pixel intensity distribution (i.e.spectral features).
- Foreground aperture: When areas of a moving object are coloured uniformly, changes in these areas may not be detected. As a result, the target object may not appear as the foreground, whose mask contains a false-negative test.
- Moved/inserted background objects: a new background object appeared in the background can be easily misclassified as foreground.
- Dynamic backgrounds: The background can fluctuate, so it needs a model that can represent disjoint sets of pixel values, such as swaying trees, water ripples and water surface. In each case, there will be much error.
- Beginning moving object: When an object in the background is initially moved, the newly displayed part of the object and the background is detected, which is called "ghosting".
- Shadow: Since shadow moves with a foreground object, it is easily classified as part of the foreground object. However, in many applications, shadows are regarded as "noise", which decreases the model performance. Shadow detection itself is a research area [82].

In the following section, several solutions will be illustrated corresponding to those listed issues.

2.2.2 Traditional Methods

How to effectively analyse and understand the features of an image is a relevant and active research field. The objectives can be in different scales, e.g. pixels [39], blocks [40] or clusters [40]. However, they are less resistant to noise than block-based or clustering methods.

This subsection refers to the following survey about traditional methods for more details.

Most traditional remote sensing target detection methods follow a two-stage detection paradigm: candidate extraction and target verification. In the candidate extraction stage, some commonly used methods include methods based on grey value filtering, methods based on visual saliency, methods based on wavelet transform, and methods based on anomaly detection. The methods above are similar because they are unsupervised and often fail in complex environments. During the target verification phase, some common functions include Histogram of Oriented Gradients(HOG) [41, 42], Local Binary Pattern (LBP) [43], Scale-invariant Feature Transform (SIFT) [44, 45], *etc.* In addition, there are other methods that follow the sliding window detection paradigm.

As seen in the above content, the choice of feature is essential, and the texture features are more robust to illumination changes and shadows. What is more, from massive amounts of multispectral remote sensing data to microscopes, the texture is an import feature of many types of images. Hence, as one of the main problems in texture analysis, texture classification has been a long-term research topic. There are also many applications of texture classification among medical image analysis and understanding, object detection, content-based image retrieval and document classification.

As a classic pattern recognition problem, it is generally believed that the extraction of powerful texture features plays a relatively important role. Because once the weak features are used, even the best classifier will not be able to obtain proper recognition results. Therefore, most researches on texture classification have focused on the feature extraction part. Many researchers have developed texture feature extraction methods [46–48]. From these surveys, how to balance two competing goals:

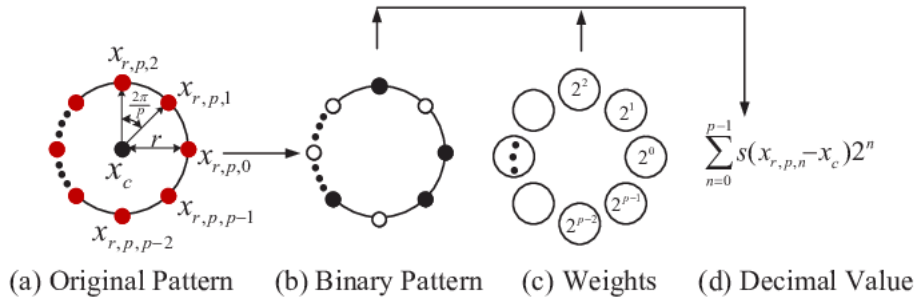


Figure 2.4: A typical LBP calculation progress. [50]

high-quality description and low computational complexity is an inherent difficulty in extracting powerful texture features. High-quality descriptors must deal with the compromise between uniqueness due to multiple textures and robustness due to a large number of intra-group changes due to changes in lighting,m rotation, scaling, blurring, noise and occlusion. The balance between high-quality descriptors and low-dimensional representations all the entire application task to run in real-time.

Local binary pattern

LBP [49] emerged as a popular feature descriptor used for still images classification. It attracted great attention due to its outstanding advantages: easy implementation, robust to illumination changes and low computational complexity.

Although initially proposed for texture analysis, LBP method has proved its efficiency in many applications, including dynamic texture recognition, remote sensing, fingerprint matching, visual inspection, image retrieval, biomedical image analysis, facial image analysis, motion analysis and edge detection [47].

The original LBP operator characterises the spatial structure of a local image patch, by encoding the difference between the pixel value of the centre point and the pixel values of its neighbouring pixels. The decimal value of the resulting binary pattern is then labelled for the given pixel. Formally, as shown in Fig 2.4, given a pixel $x_c$in an image, the LBP response is calculated by comparing the values of the LBP response with the values of its neighbouring pixels $\{x_{r,p,n}\}_{n=0}^{p-1}$, r indicating the sampling radius and n the nth sampling point of p samples in total, and the distribution of this pixel is evenly distributed in a circle centred on x_c as:

$$LBP_{r,p}(x_c) = \sum_{n=0}^{p-1} s(x_{r,p,n} - x_c)2^n, \quad s(n) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.2.1}$$

where $s()$ is a distance function. If the coordinates of x_c are $(0, 0)$, the coordinates of $x_{r,p,n}$ are given by $(-r\sin(2\pi n/p), r\cos(2\pi n/p))$. If the results are not integers, i.e. the points may not appear on the image matrix, the value of $x_{r,p,n}$ can be interpolated to estimate the greyscale values of their neighbours that belong to the image matrix.

Therefore, the texture image can be characterised by the probability distribution of the 2^p LBP pattern. The LBP operator has been extended to multi-scale analysis to process any radius and number of pixels in the neighbourhood by changing the parameters (r, p). In order to enhance the robustness of image rotation, a rotation-invariant version of $LBP^r_{r,p}i$ [83]. In this version, rotation invariance is achieved by computing and categorising the descriptor with respect to a codebook by a clustering method on the learned binary codes. The proposed method not only maintains the complete structural information extracted by LBP, but it also captures the complementary information by utilizing the magnitude information, thereby achieving greater discrimination.

The author of LBP observed that some LBP patterns represent basic texture microstructures, and named these patterns as uniform patterns with an U value up to two [38] defined as

$$U(LBP_{r,p} = \sum_{n=0}^{p-1} \mid s(x_{r,p,n} - x_c) - s(x_{r,p,mod(n+1,p)}) - x_c) \mid,$$

this way, $U(LBP_{r,p})$ will calculate a bitwise conversion from 0 to 1 and vice versa. The unified descriptor $LBP^{u^2}_{r,p}$ has $p(p-1)+3$ categories, including $p(p-1)+2$ different unified patterns and a non-uniform group containing all non-uniform patterns. Ojala *et al.* [38] propose to further group the uniform patterns into $p+1$ different radiation-invariant categories, resulting in a rotation-invariant unified

descriptor LBP with a lower dimension $p + 2$:

$$LBP_{r,p}^{riu2} = \begin{cases} \sum_{n=0}^{p-1} s(x_{r,p,n} - x_c), & if U(LBP_{r,p}) \leq 2 \\ p + 1, & otherwise \end{cases} \tag{2.2.2}$$

The LBP is simple in computation and low in complexity to avoid memory and processing time limitations. What is more, the LBP descriptor has capabilities to deal with some different applications, such as illumination changes, scaling and bur problem. However, the main limitation in LBP is that it is very sensitive to the noise with the central pixel as a threshold value. To overcome the weakness of LBP in noise with the central pixel, Local Ternary Pattern [51] is the first method to prove itself robust to noise with central pixel by introducing manual threshold value. However, it is challengeable to set the right value for getting high performance. To deal with this issue, some noticeable extensions have been proposed, such as Improved Local Ternary Pattern (ILTP) [52] and Local Adaptive Ternary Pattern (LATP) [53], which automatically set a threshold value for every region. Besides, Complete Local pattern (CLBP) [43] and Complete Local Ternary Pattern (CLTP) [51] consider the global features rather than local features to get more accurate information about patterns, because sometimes above descriptors cannot generate distinguished features between two different grey value-based local features of the image.

2.2.3 Semantic Segmentation with Neural Network

While the performance of handcrafted feature extraction approaches was saturated after 2010, the great success of convolutional neural networks has drawn attention from conventional methods to deep learning-based models. Apart from image classification, deep convolutional architectures have proved their efficiency on a number of computer vision tasks, including semantic segmentation, object detection, video analysis, *etc.* In general, semantic segmentation is one of the high-level tasks that paves the way for scene understanding. The fact that more and more applications are getting nutrition from image knowledge underlines the importance of scene understanding as a core issue in computer vision.

Terminology and background concepts

In order to correctly understand how deep learning architecture handles semantic segmentation, it is important to know that semantic segmentation is not an isolated field, but a natural step from coarse to fine inference. Initially, it was designed for image classification tasks; later researchers found that the stacked convolutional layers were capable of learning discriminative and highly abstract representations. Thus it was modified and adapted to tackle other problems such as semantic segmentation. Location or detection is the next step towards fine-grained inference, providing both categories and spatial locations of the target objects. Semantic segmentation is a natural step to achieve fine-grained inference. The goal is to make dense predictions to infer the labels of each pixel. Further improvements can be made, such as, instance segmentation that separates labels for different instances of the same category, or even part-based segmentation that low-level decomposition of the divided categories into their constituent parts. Figure 2.5 shows examples of the aforementioned applications.

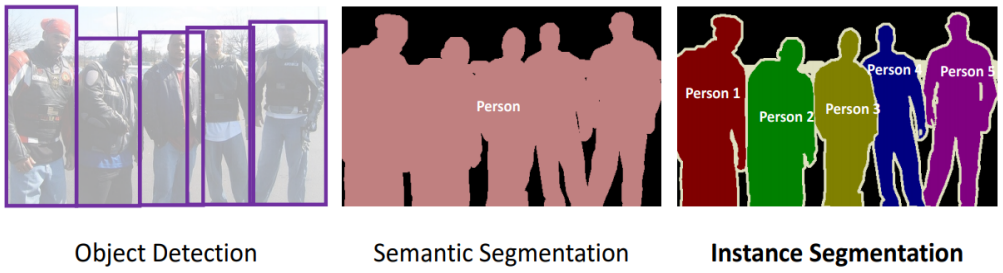


Figure 2.5: Evolution of object recognition: Object detection vs Semantic segmentation vs Instance segmentation

Common deep network architectures

As mentioned earlier in Chapter 2.1.2, some deep networks have contributed so much to the field that they have become well-known standards. It is worth reviewing the evolution of the architecture design and several milestones, some of which still regarded as the first choice for laboratory test or prototype models in practical applications. They have made significant contributions to the field of computer vi-

sion, as they are often used as the basis commonly known as 'backbone' for semantic segmentation systems:

1. **AlexNet**: the deep CNN that won the 2012 ImageNet competition with a TOP-5 test accuracy 84.6%, presented by Krizhevsky *et al.* [54]. The architecture is straightforward. It consists of 5 convolutional layers, each followed by a max-pooling layer and a Rectified Linear Unit (ReLUs), and three fully connected layers, the first two of which are followed by a dropout layer respectively.

2. **VGG**(Visual Geometry Group): the model that won the 2013 ImageNet competition with a TOP-5 test accuracy 92.7% (Figure 2.6) [67]. It has different versions named after the number of weighted layers (convolutional layers and fully connected layers), e.g. VGG-16 has 16 layers. The VGG uses a stack of convolutional layers with a smaller receiving field instead of several convolutional layers with a larger receiving field. This design results in fewer parameters and more non-linearity in between, making the decision function more discriminative and the model easier to train.

3. **GoogLeNet**: is a compound CNN architecture, introduced by Szegedy *et al.*, which won the 2014 ImageNet competition with a TOP-5 test accuracy 92.7% [68]. The GoogLeNet consists of 22 layers and a newly introduced building block called inception module, see Figure 2.7. The inception block is designed for expanding the network depth and width without increasing the number of parameters by introducing 1×1 convolutional layers for dimension reduction, which massively reduces the number of feature maps at minimal extra cost.

4. **ResNet**: is especially remarkable not only due to its success in the 2016 ImageNet competition with a TOP-5 test accuracy 96.4%, but also because of the fact that its incomparable depth (152 layers) and the introduction of residual blocks [66](Figure 2.8). The ResNet belongs to Microsoft's model [66]. He *et al.* effectively solves the vanishing gradient problem by introducing an

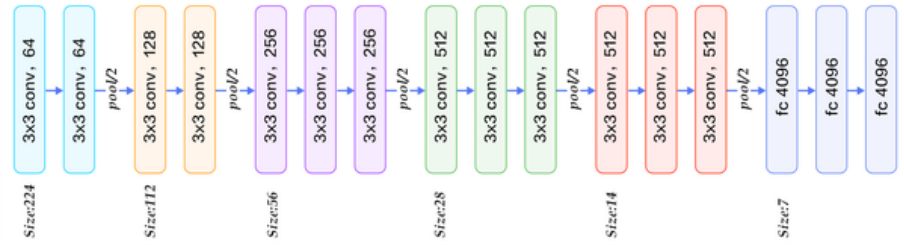


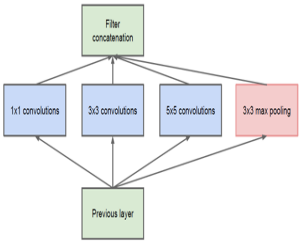
Figure 2.6: VGG16 Architecture

identity shortcut connection (skipping layers with a confident stride, behaving as if the depth of the network is automatically selected for different scale of problems). They proposed a mutation residual block before activation in which the gradient can easily flow through fast connections without being hindered during the back-propagation.

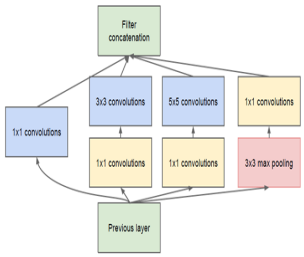
5. **DenseNet**: is compelling due to its fewer parameters and high accuracy, comparing with ResNet. The DenseNet, jointly invented by Cornwell University, Tsinghua University and Facebook AI Research, won the best paper award with over 8000 citations in 2017 CVPR. Compared with ResNet that applies to skip connection to the beginning and end of each block unit, in DenseNet [77] each layer takes all preceding feature maps as input. Thus it appears to be more parameter efficiency (Figure 2.9). What is more, the DenseNet architecture can clearly distinguish the information added to the network from the retained information. Concatenation is used. Each layer receives "collective knowledge" from all previous layers, and the network can be thinner and more compact, that is, the number of channels can be less. However, the back-propagation algorithm for tuning network parameters requires to store outputs from every layer, thus the dense connection yield more consumption to memory on GPU during training.

Semantic segmentation approaches

For tasks like image classification, deep learning models are capable of extracting highly abstract feature maps as the layers proceed and the receptive field being



(a) Inception module, basic version [68]



(b) Inception module with dimension reductions [68]

Figure 2.7: Inception module [68]

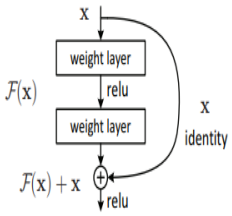


Figure 2.8: Residual learning: a building block [66]

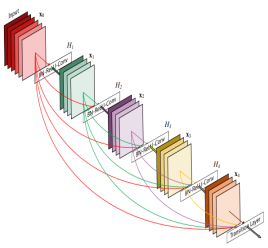


Figure 2.9: Dense block [77]

expanded, eventually the top convolutional layers will produce a number of class-specific feature representations, which are used by final fully connected layer(s) to make class predictions. On the other hand, semantic segmentation requires pixel-level classification, and the common approach is to take the convolutional layers of a well-trained image classification model and concatenate a few more "up-sampling" layers to equal the resolution of an input image and output prediction mask. The following section briefs the development of deep segmentation models.

Weakly and Semi-supervised learning for semantic segmentation using deep models

Conventional supervised deep learning models require specially labelled data to achieve specific training purposes. Often data labelling is costly, sometimes even require expert knowledge.

Papandreou *et al.* [55] proposed a weak semi-supervised learning method using weak annotations, which could be used alone or in combination with a small number of robust annotations. They implemented a method called Expectation-Maximization (EM) for training DCNNs from weakly annotated data.

At the same year, Hong *et al.* proposed a semi-supervised method (Decoupled-Net), which used two separate networks, one for classification (classifying object labels) and other for segmentation (getting graphic background segmentation for each classified label) [56]. Later in 2017, Luo *et al.* proposed a weak and semi-supervised dual image segmentation (DIS) learning strategy that performed segmentation (capturing certain object categories) and reconstruction (perfect object shapes and boundaries). The idea was to predict labels, label the label map from the input image, and perform image reconstruction using the predicted label map. Barnes *et al.* developed a weak supervision method for autonomous driving applications, which was used to generate a large number of the labelled image (from multiple sensors and data collected during driving). The labelled images only contained route suggestions without any human annotations. In 2018, Wei *et al.* proposed a weak semi-supervised method using multiple dilated convolutions [70]. They proposed an enhanced classification network with multiple convolutional (MDC) blocks that generate dense object localizations maps and use them for semantic segmentation

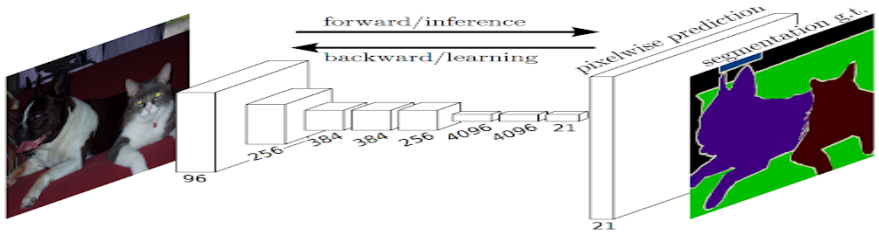


Figure 2.10: Fully convolutional networks [65]

in weakly- and semi-supervised ways.

Weakly and semi-supervised learning aims to reduce the burden of full annotations. These methods use weak annotations of image-level labels (information about which object classes are present) and bounding boxes (rough object locations) to improve learning performance.

Fully convolutional networks for semantic segmentation (FCN)

The paper presented by Long *et al.* [65] is regarded as the pioneering work for fully convolutional image segmentation. They proposed an architecture that can capture images of any size and generate output with corresponding spatial dimensions. The authors claimed that by replacing the fully connected layer(s) of a deep classification model with one or more convolutional layers, the network can output a heatmap that shows the prediction confidence on each pixel location, which is eventually turned into a segmentation class map (e.g. labelling foreground objects as 1, 2, ..., background as 0).

The key idea of the FCN-based method [62, 63, 65] shown in Figure 2.10 is that they can learn the mapping from pixels without extracting region suggestions. The FCN network pipeline is an extension of the classification networks. The main idea is to make classic CNNs accept images of any sizes as input by removing the fully connected layer(s), which constrains the dimension of the inputs as the input dimension of fully connected layer(s) is always fixed. In contrast to them, FCNs only have convolutional and pooling layers, which allows them to make predictions on inputs of any size.

One problem in FCN method is that when propagating multiple alternating convolutions and pooling layers at the feed-forward stage, the resolution of the output feature map will be down-sampled, while only the high-level feature maps (com-

monly eight times smaller than the input image) are upsampled. Therefore, the output prediction mask of FCN would have relatively blurred boundaries. Recently, several variation forms of FCN have been proposed to tackle this problem. For example, Eigen and Fergus [64] proposed a multi-scale convolutional network consisting of multi-scale subnets with different resolution outputs to improve the coarse prediction gradually. In the FCN paper [65], the most advanced architecture combines coarse high-level information with fine low-level information, where the multi-layer output is followed by a deconvolutional layer with bilinear upsampling for pixel-level prediction at a similar scale as the input image. In order to accurately reconstruct the highly non-linear object boundary, a deep deconvolution network was used to identify pixel-level class labels and predict segmentation masks [57]. In addition to the deconvolution layer, DeepLab-CRF [58, 59] also provides another method to improve the output resolution that atrous spatial pyramid pooling (ASPP) is exploited for segmenting objects at multiple scales to improve locations of object boundaries basing on DCNNs.

Apart from trying to improve output resolution, others attempted to improve segmentation accuracy by using contextual information. For example, the global average pooling is used to obtain the global context. The global context is added to the fully convolutional network for semantic segmentation, which brings continuous network for semantic segmentation, which brings continuous improvement in accuracy [61]. As its extension, the pyramid scene analysis proposes more representative the global context information through context aggregation based on different regions [69]. As an alternative to the global context, exponentially expanded dilated convolutions [70] are used to aggregate multi-scale content information. Lin *et al.* explored two types of spatial contexts to improve segmentation performance: patch-patch context and patch-background context. The two types explicitly modelled context relationships using CRF [71].

Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation

Following the idea of FCN as mentioned above, Tiramisu [60] extends DenseNets for segmentation purpose by adding an upsampling path and downsampling path to recover the full input resolution. The downsampling path has two down-conversion

(TD), while the up-sampling has two up conversions (TU). In the FCN paper, the authors introduced three variations: FCN-32s, FCN-16s, FCN-8s, indicating 32, 16 and 8 times upsampled output respectively, and the reason for the decreasing upsampling ratio is that the prediction outputs combine feature maps of different scale. Thus they are able to preserve both high-level and low-level information. The authors of Tiramisu proposed to establish a connection between the bottom-level feature maps and top-level outputs. Since all the convolutional layers in DenseNet are connected, by concatenating high-resolution feature maps with a large number of input filters from all layers below, it will lead to an explosive growth of parameters. For mitigating this effect, the tiramisu architecture only upsamples the feature map created by the previous dense block, which allows multiple dense blocks at each resolution of an upsampling path, regardless of the number of pool layers. Besides, given a network architecture, up-sampled dense blocks combine information contained in other dense blocks of the same resolution. High-resolution information passes through the standard skip connection between the downsampling path and the upsampling path.

DeepLab

In this part, DeepLab v1 [58] and DeepLab v2 [59] are reviewed together because they both use Atrous convolution and fully connected conditional random fields (CRF). DeepLab v2 also has another technology called Atrous Space Pyramid Pooling (ASPP), which is the main difference from DeepLab v1. Besides, other than using VGGNet the same as DeepLab v1, DeepLab v2 also tested the performance when using ResNet as the backbone network.

Before reviewing the details of DeepLab series models, a few concepts/submodules are explained below for supplemental information:

1. **Fully connected conditional random field (CRF)**: is used for post-processing, which can combine low-level image information with the output of a multi-category inference system that produces a category score for each pixel. It is known that the invariance inherent in the spatial transformation of the CNN architecture limits the spatial accuracy of the segmentation task. One possible and common way refines the output of the segmentation system and

enhances its ability to capture fine-grained detail is to apply post-processing stages using CRF.

2. **Atrous convolution** is also called "dilated convolution", which is commonly used in wavelet transform and is now used in deep learning convolutions. Atrous convolution allows the network to expand the filter's field of view to include a larger context without extra cost. Therefore, it provides an effective mechanism to control the field of view and find the best balance between precise positioning and contextual assimilation.

3. **Atrous spatial pyramid pooling (ASPP)** is an atrous version of SPP, which is proposed by Chen and his team [59]. In that paper, a network architecture called SPPNet takes advantage of the SPP structure for multi-scale feature learning, such that the model has robust performance dealing with objects from different scales. In ASPP, it consists of multiple parallel atrous convolution layers with different sampling rates, which can strongly segment objects at multi scales.

The DeepLab v1 overcomes weak localization property of deep networks by combining the responses at the final re-purposed and finetuned VGG-16 layer with a fully connected CRF. TThe DeepLab v2 is a residual net variant of DeepLab v1 by adapting the ResNet image classification, which can achieve better semantic segmentation performance when compared with the original DeepLab v1 model based on VGG-16. What is more, DeepLab v2 applies ASPP to produce semantically accurate predications and detailed segmentation maps along object boundaries.

DeepLabv3

Later, Chen *et al.* [78] revisited random convolution and proposed a new system network called DeepLab v3. It is different from the previous two versions, DeepLab v1 and DeepLab v2. The Deeplab v3 owns a new module in which atomic convolution works in cascade or parallel model to capture multi-scale contexts by using multiple atomic rates and uses batch normalization for training. The main idea is to copy several copies of the last block in ResNet and arrange them in a cascaded manner. Compared with the conventional convolution with a larger filter, infinite convolution

can effectively expand the field of view of the filter without increasing the number of parameters or the amount of calculation. Expanded convolution is a simple and powerful alternation to deconvolution in intensive prediction tasks.

Later, DeepLab v3+ [79] is introduced, which is an extended version of DeepLab v3. Inspired by Alvarez *et al.* [80], the authors propose a decoder module in which the encoder features are upsampled by four instead of sixteen in DeepLab v3 and then are connected to the corresponding low-level features from the network backbone with the same spatial resolution. They adopted the Xception model [81]. They applied depth-wise separable convolutions to reduce the computational complexity to the ASPP and decoder modules.

SegNet:A deep convolutional encoder-decoder

In addition to the FCN architecture, other variants have been developed to convert the network, the purpose of which is to classify the network to make it suitable for segmentation. It can be said that FCN-based architecture is more popular and successful, but other alternatives are also noticeable.

SegNet is an obvious example of these alternations. Badrinarayanan *et al.* [76] proposed an encoder-decoder structure for deep fully convolutional neural networks, called SegNet. The encoder network has the same topology as VGG-16 [67], without fully connected layers, followed by the decoder network from for pixel-by-pixel classification. By using the decoders that each decoder corresponds to each encoder, SegNet can achieve a higher resolution than in FCN [65]. The key function of SegNet is that the information transfer direction is direct, not convolution. When dealing with image segmentation problems, especially scene segmentation tasks, SegNet is one of the best models.

U-Net

Unlike semantic segmentation for natural scene images that have an available public dataset with well-labelled image data, e.g. dataset for PASCAL Visual Object Classes (VOC) challenge [84] and Microsoft COCO dataset [85], semantic segmentation for biological microscope images may only have dozens of labelled data at one time since labelling medical images requires expert knowledge. Reonneberger *et al.* proposed a network architecture called U-net, which consists of two parts: a con-

traction part for learning feature representations and an expansion part for spatially reproducing the detail patterns on the same scale. It is named U-Net for its shape since the architecture is generally symmetric with large feature maps at the top bottom of the model and small feature maps in the middle. It is also called encoder-decoder structure, where the down-sampling process encodes highly abstract global information into smaller scale feature maps, while the up-sampling process decodes the information and gradually recover the details such as object boundary and thin edges. Further, by concatenating two feature maps of the same scale level in the corresponding layer of encoder and decoder (in the paper they describe it as establishing skip connection between those two layers, same mechanisms as the identity mapping in ResNet), the lower-level feature maps are very well preserved, plus the training speed is boosted. Thus it outperforms other methods when segmenting fine details such as cells and vessels in the biomedical field, while massively reduces the size of the training set.

The idea of concatenating bottom layer low-level feature maps with top layer feature maps to generate final dense maps are widely adopted in the tasks that have a high requirement for fine details and subtle features. The whole network is compact and easy to train. However, sometimes it also preserves a few noise from the background, especially when those noises from the background have a similar pattern as the learned foreground features. Compared with other fully convolutional segmentation networks, the main contribution of U-Net in this sense is that when performing upsampling and in-depth research in the network, the author associates features of lower resolution with features of downsampling in order to better locate and learn to follow the convolution. Since upsampling is a spare operation, we need to prepare well from earlier stages to represent localization better. A similar idea of combining comparable levels has also been found in Functional Pyramid Network (FPN) [73].

FastFCN:Rethinking Dilated Convolution in the Backbone for Semantic Segmentation

This paper proposed a joint upsampling module named Joint Pyramid Upsampling (JPU) to replace the atrous convolutions (in this paper it is equivalently called

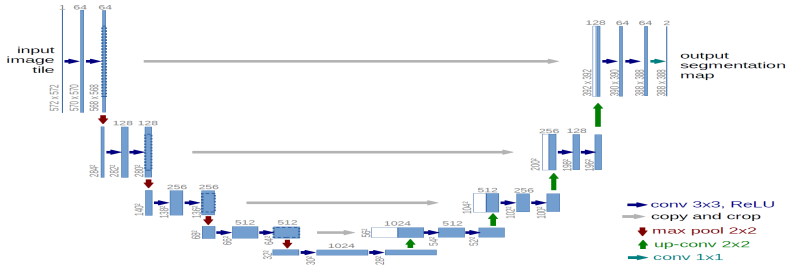


Figure 2.11: U-Net: Convolutional network for biomedical image [72]

dilated convolution). This work dramatically reduces the computational complexity of the network on the promise of ensuring network accuracy. The backbone of FastFCN is identical to FCN, which is followed by a JPU module. Eventually, the author gets a network which has comparable performance but improved speed on many tasks [74].

This article made a small modification to the dilated convolutions in the FCN network to achieve a more consistent result. The paper is easy to understand and apply. However, it contains some non-essential works because it may also be due to the reason of too simple. In short, for engineering, it is better to get a faster FPS network.

Gate-SCNN: Gated Shape CNNs for Semantic Segmentation

Takikawa *et al.* [75] carefully designed the Gated Convolution Layer (GCL) structure and used local supervision to let the shape stream only process the boundary information. The authors argued that generally semantic segmentation deals with colour(RGB) images, where the colour channels that encrypt different information should be processed separately, while the existing CNNs all treated them uniformly, thus causes information loss. For example, colour and texture contain some relatively low-level information. However, these pieces of information may be not as same important as the outlines for semantic segmentation.

Hence the authors proposed a new, two-stream CNN, which will remain explicitly independent the shape information into a processing branch. The two-stream CNN contains a traditional CNN stream and the other shape stream CNN. What is more, the Gated-SCNN uses a new gating mechanism to connect the intermediate layers. The fusion of information between steams is done at the very end through the

fusion module. The paper also exploits a new loss function to predict high-quality boundaries that encourage the predicted semantic segmentation masks to align with ground-truth boundaries.

In this thesis, we first use the conventional LBP method to set our baseline, because the subject "watermarking retrieval from 3D printed objects" is still virgin land. With further research, we import CNN as our methods. The FCN-based method is firstly used because it can learn the mapping from pixels without extracting region suggestions and is desired for local tasks. Further, as our study more focuses, we combine ResNet50 and U-Net based network in Chapter 7 to explore the effect of the illumination model on the generation of synthetic training datasets.

2.3 Synthetic dataset creation and domain randomization

2.3.1 Data Augmentation

Having sufficient training data is critical for training a robust deep learning model. When acquiring data becomes a challenge, a common way is applying data augmentation during the training phase, which also reduces overfitting. On the other hand, researchers have found that using supplemental synthetic data to expand the training set is efficient and at low cost, mainly because it is much easier to obtain accurate ground truth label along with the synthetic images, plus it is easy and inexpensive to obtain synthetic images with environmental variations, such as view angle, shadow, illumination, texture, and background, to some extent simulating the real-world scenario, thus increasing the model robustness.

Generally, there are two types of data expansion strategies for training models [86]. One is to perform all necessary conversions in advance, thereby significantly increasing the size of the dataset. Another option is to perform these conversions before converting small batches to machine learning models. The first option is called offline augmentation. For relatively small datasets, this method is the preferred method because we can eventually increase the size of the dataset by a factor

equal to the number of performed conversions. For example, by flipping the whole images, we can increase the number of datasets by two times. The other option is called online augmentation. For large scale datasets, this method is preferred because it would be challenging to deal with explosive data growth. Instead, we have to perform a transformation on the mini-batch input the model. Some machine learning frameworks support online expansion, which can accelerate online expansion on GPU [102].

Because our dataset is small, we will only give more technical details about offline data augmentation.

Popular Basic Augmentation Techniques

In this part, we introduce some commonly used augmentation techniques. Before discussing these techniques, we assume that we do not need to consider the problem of exceeding the image boundary for simplicity.

Flip: There are two flipping methods, vertical and horizontal. Generally, the horizontal flip is more common than a vertical one because some frames do not provide vertical flipping functions. As one of the most natural methods to implement, both flipping functions have been proved useful for datasets such as CIFAR-10 [103] and ImageNet [104]

Crop: It can be fixed position crop, e.g. central crop, that we keep a fixed size on the central area of the image and discard the rest, or we randomly select a window from the original image, then we resize this window to the original image size. The latter method is usually called "random cropping".

Rotation: Rotation augmentation can be achieved by rotating the image for a fixed or random angle (0° - 360°). There are two things to note. One is that the improvement of rotation safety depends mainly on the rotation parameters. For example, slight rotation, between 1 to 20 or -1 to -20, may be useful for digital recognition tasks, such as MNIST. However, as the degree of rotations increases, the converted data labels will no longer be retained. The second thing is that the rotated image size may not be retained. Rotating an image at a small angle will change the final image size, but if the image is square, rotating it to a right angle

will preserve the image size. If it is rectangular, rotating it 180 degrees will retain the size.

Translation: The translation is a beneficial conversion, which can reduce the effect of positional shifts in the data. In other words, this augmentation can force a convolutional neural network to look for position invariant features, because many objects can be located almost anywhere in the image.

Colour space transformations: The image data is encoded into stacked matrices, each of which has a size of height × weight. These matrices represent the pixel values of single RGB colour value. Illumination bias is one of the most common challenges in image recognition problems. Therefore, the effectiveness of colour space conversion, which is also called photometric conversion, is very intuitive in concept. A fast fix processing of images that are too bright or too dark is to traverse the image and decrease or increase the pixel value by a constant value. One common colour space operation is to stitch RGB colour matrices. Another transformation includes limiting the pixel value to a certain minimum or maximum value. The internal representation of colours in digital images makes them suitable for many augmentation strategies. By increasing the colour space, we can create new RGB colour images freely. Changing the colour distribution of the image can well solve the lighting variance of the test data.

However, similar to geometric transformation, the disadvantage of colour space transformation is increased memory, transformation cost and training time. Besides, the colour conversion may discard important colour information. Hence, the colour space transformation will be beneficial to the spatial characteristics, which can eliminate the colour deviation existing in the dataset.

Noise injection: Overfitting usually occurs when a neural network tries to learn high-frequency features that may not be useful. The noise injection involves injecting a matrix of random values derived from a Gaussian (or other types of) distribution. Moreno-Barea *et al.* [87] conducted a noise injection test on nine datasets in the UCI Information Library [86]. Adding noise to the image can help CNN learn more robust features than no-noise images.

Random erasing: Random erasing is another data augmentation technique,

proposed by Zhong *et al.* [88]. Inspired by the dropout mechanism, this technology is specially designed to meet the image recognition challenges caused by occlusion. Random erasing will prevent occlusion by forcing the model to learn more characteristic features about the image. Thereby, it can prevent the model from overfitting to a certain visual feature in the image. In addition to the visual challenge of occlusion, random erasing can also ensure that the network to learn features from all aspects, not overfitting to conspicuous local features.

Random erasing is a data augmentation method designed to prevent overfitting by changing the input space directly. By deleting certain input patches, the model is forced to look for other descriptive features. It can also be stacked on top of other augmentation technologies, such as horizontal flips and colour filters. However, one disadvantage of random erasing is that it does not always preserve the label information, such as in handwritten digit recognition. Therefore, depending on the dataset and task, some manual intervention may be required.

A note on combining general augmentations: In the extensions discussed, geometric transformations, colour space transformations, kernel filters, mixing images, and random erasing, almost all of these transformations also have associated distortion amplitude parameters. There are many potential expansions, and the size and space almost continue, so it is easy to conceptualise the enormous expansion of the augmentation search space. The combined data augmentation methods, such as flipping, colour transformations and random erasing, may cause data size to expand significantly. However, this cannot be guaranteed to be advantageous. In domains with minimal data, this may lead to further overfitting. Therefore, it is necessary to consider the use of search algorithms to derive an optional subset of enhanced data from training deep learning models.

Advanced Augmentation Methods

Conventional image synthesis methods generate 2D or 3D digital images from a descriptive scene file using a rendering program. Since Goodfellow *et al.* [89] proposed Generative Adversarial Network (GAN) on 2014, researchers have gradually exposed the immense potential of this adversarial training mechanism for training

deep neural networks to generate 1D (sound) or higher dimension signals (images) that can fool the human visual system. One of its successful application scenarios is domain adaptation, that the GAN model modifies synthetic images to the real-world domain, which images are used as supplemental training data, such that the discrimination model is less likely overfitted to the synthetic data.

In many studies, as above shown, these classic augmentation methods have been shown to improve the performance of image data. New methods that look promising are also being studied. These methods include Adversarial Training [89, 90], Adversarial Network Generation [91–93], Style Transfer [94], and Reinforcement Learning [95] to search through the space of augmentation possibilities.

Adversarial training: is a fascinating topic in deep learning. Apart from being a crucial component in adversarial network, it can also be used to learn noise augmentation, which can help the model perform better. Goodfellow *et al.* [89] firstly induced to perform an adversarial attack by maximising the distance of feature maps while minimising visual changes on the image data, such that an image of panda (by humans) is classified as gibbon by the model. On the other hand, the images generated by adversarial training can help the model to learn more powerful features that are not affected by noise distortion. Miyato *et al.* extent the method to unlabelled samples [90]. Adversarial images can be added to training data for improving the robustness of the adversarial examples. It also improves the accuracy of the classifier and can be seen as a form of data augmentation. However, because the adversarial learning is limited to small changes, this method cannot make full use of the data augmentation features to maintain the appearance of the image.

Adversarial Network Generation: The main problem with this method is that it is difficult to generate high-resolution output images with GAN, due to graphic memory limit. It turns out that using GAN for data augmentation is valid for MNIST and CIFAR-10, whose images are often resized to 224×224. However, generating a higher resolution output than this requires very complex networks and expertise.

Mirza *et al.* and Odena *et al.* used generative models that generate an image conditioned on their class, at pixel level with a convolutional encoder-decoder ar-

chitecture, which could be directly used to expand the dataset. However, based on GAN, Antoniou *et al.* are the first one who directly used it for data augmentation [96]. In this case, the author can make adjustments directly on the given images. However, this method seems to produce sub-optimal results because the generation has nothing to do with classification. Salimans *et al.* train a discriminator coupled with the classifier by adding an extra class to the classifier to generate images. In this case, the unlabelled data generated by the generator can be seen as a form of data augmentation. Methods such as triple GAN [97] and Bayesian data augmentation [98] proposed to train the classifier and generator together. However, these models are based on generating samples directly from noise. It seems to be much more complicated than converting a given image, especially when the training data is reduced [86].

Most of the presented GAN model is designed for semi-supervised and unsupervised learning. The progress seems very costless. However, it is not friendly for small datasets, and we can hardly control generated output images, even using conditional GAN method.

Style Transfer: is another exciting topic to prevent overfitting in deep learning. The working principle of transfer learning is to train the network on a large dataset, and then the weights remains unchanged, while the input image pixel values are tuned by the optimiser in the new classification task. The loss function is derived from network to convert the style of one image to the other, while preserving the original attributes, such as outlines, shadows, lines, strokes and areas.

Style transfer expands the scope of light changes and can also encode different textures and artistic style. However, one disadvantage of style transfer augmentation method is that this method needs to select a mode to transfer images, which means most classical texture transfer methods rely on resampling texture for a given content image [99].

Reinforcement Learning for Data Augmentation: is an existing field to explore the possibility of reinforcement learning strategies. In reinforcement learning algorithms, strategies are similar to those of learning algorithms. The strategy determines what measures to take to achieve specific goals in a given state. The Au-

toAugment approach includes many sub-strategies, and each sub-strategy contains an image transformation and transformation amplitude [100].

Rendering synthetic imagery for Data Augmentation: provides the promise of a perfect label beside the data. The rendering engines can offer photo-realistic visual images by finely adapting camera pose, lighting location, temperature, distribution, and object geometry.

Currently, we assert that the significant factor that limits the success of many computer vision algorithms is the lack of label data and the accuracy of the labels provided. Especially for viewpoint estimation, the space of possible angles is ample, and it is challenging to collect enough samples for each angle. Also, it is challenging for human annotators to mark each image with the real ground angle accurately. We can use the first four advanced methods for data augmentation. However, by using the rendered photo-realistic data, we can control the viewpoint deviation and can create a uniform distribution. We can also adequately sample lighting conditions and occlusions of other objects. The subsection will give more details about photo-realistic rendering for visual learning.

2.3.2 The review of physically based synthetic image for machine learning

The state-of-the-art synthetic image generation for machine learning

The researchers of computer vision have built various image datasets for decades. As long as they have been building these datasets, they have been fighting data biases. Despite that digital data is growing explosively, most of them (more than 75% according to [101]) remain unstructured, let alone labelled for specific purposes since building high-quality labelled datasets from large amounts of data is time-consuming and expensive. Hence, researchers continue to look for better ways to annotate large-quality datasets. For example, the computer vision community has used online labour markets (such as Amazon's Mechanical Turk) to crowdsource simple tasks, such as image-level labels or border annotations. However, the annotators often lack the knowledge to complete certain types of tasks [105, 106]. Evenly, answers

from the labour market are not consistently the same as each other. The answers from experts become the most crucial, while those from unskilled workers tend to introduce noise (errors) to the dataset. However, experts are rare in a population and not adequately identified [107].

The problem detailed above provides compelling arguments for the automatic collection and labelling of datasets. The usability of 3D models has dramatically increased. In contrast, the cost of obtaining them has dropped, which provides an attractive way to explore: the rendered images can be customised for many computer vision applications.

Sun and Saenko conducted domain adaption based on "decorrelation features" [108]. They found that the performance of detectors trained on virtual data and suitable for real-image statistics is similar to that of detectors trained on actual image datasets, including ImageNet. The performance is comparable. The result shows that non-photorealistic data is as useful as attempts to render more realistic images. Instead of domain adaption, Tobin *et al.* [157] tired domain randomisation, which is a fundamentally different method that can overcome the considerable obstacle they call the reality gap. They argued that it is impossible to include all physical effects in current simulators and that analogue sensors cannot reproduce noise behaviour in the same detail as similar sensors in the real world. Therefore, they focused on localising objects (on a table for robotic control), where they discovered that the method is sensitive to the following factors: the number of training images, textures, distractors in training data, camera positions and pre-trained weights in the detection model. However, the amount of random noise has no significant effect on the model, because a small amount of random noise can improve convergence and contribute network less sensitive to local minima during training. Randomising the position of the camera can also continuously improve accuracy. What is more, the authors proved that when the training samples are about 5000, the average error of the network has significantly decreased and is almost as small as the pre-trained one. At about 16000 training samples, the network trained from scratch exceeded the trained network in advance.

Further, **Tremblay** *et al.* [163] transferred the domain randomisation technology

to target detection and conducted further research. They placed a random number of synthetic cars in arbitrary positions and directions in the 3D scene and added a random number of geometric shapes to the scene. The combined, resulting images with the automatically generated ground truth label are used to train the neural network. Furthermore, various pre-trained general objects are used for more accurate transfer learning. Finally, compared to training only on real data, better results can be obtained by fine-tuning real data to enhance synthetic DR data.

Hinterstoisser *et al.* [109] studied the advantages of combining transfer learning with synthetic training data. The goal of their paper is to use a large amount of available synthetic data in the source domain to train a neural network to classify real data in the target domain. The generation pipeline uses not only uniform distribution to place objects at random locations in a randomly selected highly chaotic real background images, but also add random Gaussian noise to the rendered object to better integrate rendering. Hinterstoisser *et al.* indicated that the images from different cameras will lead to different results, and the object detector retrained on the synthesised data will cause poor performance. Nevertheless, on the contrary, the freeze feature extractor can always significantly improve performance. In addition, they claim that their experimental results show that simple rendering is sufficient to achieve excellent performance, and complex scene synthesis does not seem to be necessary.

Movshovitz-Attias *et al.* [110] used an extensive database of highly detailed 3D models to create a large number of synthetic images. In order to diversify the generated data, many rendering parameters have changed a lot, exceptionally light location, intensity and temperature. More complex the rendering process becomes, the fewer errors will decrease. They studied the combination of synthetic data and real data for training. They found that when using low-quality renders, once the number of renders dominates the train set, the error will increase. Overall, this is a significant improvement when replacing up to 50% of images with render data. One possible explanation provided by the authors is that the angular distribution of the actual training dataset lacks balance. From experiments, authors infer that generalising from synthetic images with a certain amount of real data can improve the

estimation accuracy [110]. Besides, the use of sophisticated materials and lighting is also an important aspect of synthetic datasets.

Latest, **Hodan** *et al.* [111] conducted a more in-depth study on the highly realistic composite images generated by physics-based rendering for training CNN-based target detectors. In the paper, on the one hand, authors use commercial Autodesk MAYA and MAX software for low, medium and high rendering quality settings for physical-based rendering. On the other hand, the synthetic data generation pipeline from Hodan *et al.* is used for "reference" image creation. It is recommended that for scenes with more straightforward lighting and materials, low-quality rendering is sufficient, which shows that giving up the quality of physically based rendering (PBR) is conducive to speed. However, even low-quality rendered images based on physics have better performance than non-PBR images. However, there are also some limitations to their theory. Hodan *et al.* can only train the most advanced object detectors on synthetic images data.

Photorealistic synthetic images applications

Most studies generate training data by rendering images from 3D scenes through more or less photorealistic rendering [109–111, 163]. When determining the most appropriate method for generating synthetic data, it is essential to consider what type of synthetic data to have. Hence, even crude or straightforward pasting of 2D objects in front of another background images is still being used. However, in addition to standard rendering software, such as Blender and Autodesk 3D Maya/ Max and its embedded ray tracing engines, such as Octane, V-Ray, Cycles and Mitsuba render, many extraordinary methods are used to obtain synthesis data.

Richter *et al.* [112] used commercial video games to generate large scale high fidelity open world scene images with corresponding pixel-accurate ground truth labels realistic images at a low cost. Their experimental results show that training semantic segmentation model with the generated game data plus 1/3 of the CamVid training set has better performance than a model trained with full CamVid training set [113], because this dataset is small, only consisting of 367 training and 233 testing RGB images (day and dusk scenes) at 360×480 resolution. Shafaei *et al.* [115]

like Richter *et al.* also used commercial video games to segment street scenes semantically. A sample is collected from the camera in the game every second. Their experiments show that the network trained with these synthetic data achieves similar test error compared to the network trained with real data for dense image classification. Also, they claim that if simple domain adaptation techniques are applied, they can get similar or even better results via using synthetic RGB colour images than real data. Following Shafaei's work, Atapour-Abarghouei and Breckon [162] take advantage of style transfer and adversarial training to span the gap between synthetic environment data from commercial video games and real-world colour images.

Beside above data augmentation methods, Peng [116] escapes the range of commercial video games to use freely available 3D CAD models to generate synthetic images data automatically. Following Peng's idea, Tobin *et al.* explore domain randomization [157], which transfers to real images by randomizing rendering in simulators via simple 3D CAD models [157]. Varol *et al.* [117] generates photorealistic synthetic images and their corresponding ground truth for human part segmentation and depth estimation. Their pipeline that generates synthetic body is created using the SMPL body model [118]. SMPL is a realistic joint model created by thousands of high-quality 3D scans, which can create realistic poseable models that cover a wide variety of human shapes and poses.

Apparently, it is cost-efficient for generating photorealistic data with pixel-level annotations, benefiting from those 3D open-source databases, software and techniques. For tasks with limited real-world training dataset, synthetic data is a proper data augmentation to advance training neural networks. Moreover, from the previous researchers' works, by improving the complexity of the scene and lighting condition, the quality of rendered synthetic data can be significantly improved.

Hence, in this thesis, we introduce synthetic data generated by Pov-Ray and Mitsuba (open source software) to train our network and explore if these synthetic data can improve our accuracy. Besides, this thesis also studies how the synthetic data under different illumination environments of render can affect retrieved information precision. More details about Pov-Ray and Mitsuba will be presented in subsection 2.5.2.

2.4 Image Processing/Graphics

Image processing is a method to perform certain operations on an image to obtain an enhanced image or extract some useful information from it. It is a type of signal processing where the input is an image, and the output can be an image or a feature associated with the image. Today, image processing has become one of the rapidly developing technologies. It also constitutes the core research area of engineering and computer science disciplines.

2.4.1 Threshold Segmentation

Image segmentation is a method of dividing a digital image into different segment sets based on specific image characteristics (such as pixel intensity, colour, texture). The primary purpose of segmentation is to simplify the representation of the image into easy-to-analyze content. Here we mainly introduce binary segmentation using image thresholding. As one of the most commonly used segmentation techniques in region-based segmentation algorithms, image thresholding works on a grayscale image and seeks an optimal threshold to separate foreground pixels from the background.

The most commonly used thresholding algorithm is the maximum inter-class variance method proposed by Otsu [166], which selects the optimal global threshold by maximizing the variance between classes. Besides, there are threshold segmentation methods based on entropy, minimum error, co-occurrence matrix, moment retention, simple statistical method, probability relaxation, fuzzy set and a combination with other methods. The advantage of the Otsu's method is the simplicity and computational efficiency, and it is good at dealing with cases where the foreground and background histograms are far apart. However, when foreground objects are relatively small, or the histograms is indistinguishable from the background, or noises damage the image that the sharp valleys of the grayscale histogram will be distorted, it would be difficult to find an optimal threshold value that well separates the background and foreground histograms. A well-known enhancement method called the two-dimensional Otsu method is proposed to compensate for those weak-

nesses. In this way, the grey value of each neighbourhood pixel and the average value of these pixels are studied to improve the binarization result of the image distorted by noise.

One-dimensional Otsu algorithm is widely used because of its uncomplicated calculation and reliability. However, the traditional Otsu hardly considers the grey level information of pixels and ignores the spatial neighbourhood information of pixels, resulting in weak segmentation effect. Qu and Huang proposed a new method based on entropy, which has a better effect than the traditional one-dimensional Otsu algorithm [138]. Zhu *et al.* proposed a stable Otsu algorithm which improves the histogram to reduce the computational complexity of one-dimensional and two-dimensional Otsu algorithms [139]. In that paper, a 2D histogram is projected onto the diagonal, and then it is applied to 2D Otsu to detect the optimal threshold. To evaluate the actual performance of the algorithm, the author applied salt and pepper noise and Gaussian noise onto the test image. The results confirmed that the proposed method is robust against salt and paper noise with improved processing speed, but for Gaussian noise, it is less effective. Liu *et al.* believed that the mechanism of the Otsu's method is identical to the K-means method in multilevel thresholding. Both are based on the criteria that minimize intra-class differences [140]. In addition, Otsu's method deploys global thresholds, while K-Means uses local thresholds. The Otsu's method requires a grayscale histogram, but the K-means does not. Both methods can produce better segmentation results, but K-means can provide more satisfactory results than Otsu. Besides, Otsu's method consumes more time and is more complicated comparatively.

Among all the variations of Otsu's methods, no single method is enough for all kinds of challenges. However, for our particular 3D watermarking subject, since one of the intermediate feature maps turns out as a clean grey level image, the original Otsu's method is capable of retrieving bright watermarking bumps.

2.4.2 Image registration

Image registration is an image processing technique to superimpose two or more images of the same scene taken from different viewpoints or at different times by

different sensors. It helps to overcome common image rotation, scaling, and skew problems when overlapping images. Image registration is a crucial step in all image analysis tasks. The final information is obtained from a combination of various data sources (such as image fusion, change detection, and multi-channel image restoration).

Due to the diversity of images to be registered and various types of deterioration, it is impossible to design a general model suitable for all registration tasks. Each model should consider not only the inferred type of geometric distortion between images but also radiation distortion and noise damage, the required registration accuracy and data characteristics related to the application. There are three main approaches to register images.

1. **Intensity-based automatic image registration** maps pixels in each image according to relative intensity pattern. This approach registers single-peak and multi-peak image pairs, as well as 2D and 3D images [141].

2. **Control point registration** enables to select common features manually. There are two beneficial application situations. One is that when compared with using automatic feature detection to detect the priority of the entire feature set, the alignment of specific feature is prioritized. The other is when the image has repetitive patterns that use automatic feature matching to provide the unclear mapping. However, manually selecting the control point pair can provide a more precise feature map for better transform to align the feature points [141].

   Control point registration can apply multiple types of transformations to moving images. Global transformations (uniformly acting on the entire image) include affine, projection, and polynomial geometric transformations. Non-rigid transformations acting on local areas include piecewise linear and locally weighted mean transformations

3. **Automated feature detection and matching** functions can detect features such as corners and spots, match corresponding features in moving and

fixed images, and estimate geometric transformations to align matched features [141].

Our research focus is on accurate 3D watermarking feature retrieval. Once the binary bump feature maps are obtained, we apply control point registration to align images and complete the full information retrieval process.

2.4.3 Connected-component labelling

Connected component labelling (CCL) (region labelling) is an algorithmic application of graph theory in which each subset of connected components is uniquely labelled. In computer vision, the connection component label is used to detect the connection area in the binary digital image, which is different from image segmentation. In other words, it focuses on labelling different objects on an image, rather than separating foreground objects from the background. When integrated into an image recognition system, the connected component tags can process various information. Other than being performed on the resulting binary image from the thresholding step, it can also be applied to greyscale and colour images. The labelled blobs generated by CCL can be counted, filtered and tracked.

There are two different ways to implement CCL, one of which is to process a single component at a time, and another is two passes. The one component at a time is fast and straightforward for implementation and understanding. It is based on the graph traversal method in graph theory. In short, once the first pixel of the connected component is found, all connected components' pixels are marked, and then go to the next pixel of the image. This algorithm is part of the Vincent and Soille watershed segmentation algorithm [142]. The two-pass algorithm [143] iterates over two-dimensional binary data with two passes: the first time, the algorithm assigns temporary labels and recording equivalents; the second time, the algorithm replaces each temporary label with the smallest label in the equivalent class.

In this thesis, we use the one-component-at-a-time method. The size of the image and the number of foregrounds is proportional to the running time of the algorithm. If the foreground cover a large part of the image, the time complexity

will be higher than the two-pass algorithm. Our confidence map is a binary image, and the foreground objects only covers a limited part. Therefore, comparing to the two-pass algorithm, by using one component method, the time complexity is lower in practice.

2.5 3D Model Design

The 3D model is used in animation, game, construction, manufacturing, product iteration and industrial design industries, and is a critical component of digital production, It is why choosing the right 3D modelling software is indispensable, which can help to achieve creativity with minimal hassle. 3D modelling software enables users to convert their designs and blueprints into realistic models and allows artists to visualize the dimensions of their architectural designs before construction is complete. Hence, finding the best 3D modelling software is a difficult task.

2.5.1 CAD software for beginners

There are many common CAD apps to help beginners design the 3D model of user's choice and then use it to print the final part at home on FDM or SLA 3D printer.

Sketch Up: is famous for visualization and planning, covering various industries such as architecture, interior design, urban planning, engineering and architecture.

The SketchUp interface is clean and beginner-friendly. Beginners can quickly grasp the basic principles and possible to create simple 3D printable models from scratch with the first few hours of learning. SketchUp is mighty in design visualization, which may be considered the best 3D modelling software for this task [144]. It is the best 3D design/3D CAD software for graphic designers, PreViz designers, and 3D printing enthusiasts.

3ds Max: is classic and well-known in the 3D modelling software suite provided by Autodesk. It is especially popular among video game developers, visual effects artists, and architectural visualization studios. Complex particle and light simulations, cloth simulation engines, and its scripting language (MAXScript) are some of the critical features beyond its 3D modelling capabilities.

As far as 3D modelling itself is concerned, 3ds Max can create parametric objects and organic objects with polygonal, subdivision surfaces, and spline-based modelling capabilities. Among other technologies, models can be created from point cloud data. 3ds Max is one of the best 3D modelling software solutions if users are willing to spend much time learning it. The best 3D modelling software (3D Design/3D CAD software), suitable for professional 3D designers and 3D modellers in many fields.

AutoCAD is as the gold standard commercial software in the field of 3D modelling software for 2D/3D CAD and drafting. It has been available as a desktop application since 1982. Since 2010 it has been sold as a model-based, web-based and cloud-based application based AutoCAD 360. It has the largest market share in the field of 3D modelling software.

The many functions of AutoCAD make it a universal tool, which has been widely used in various industries. A dedicated toolset can help users automate design and even build an entire parts library. It is suitable for architects, engineers and graphic designers in many fields.

Blender is an intermediary of professional open-source software that can be used to create animated movies, visual effects, artwork, interactive applications, video games and 3D printable models. Blender's dazzling features include 3D modelling, texturing, raster graphics editing, fluid, particle and smoke simulation, sculpting, animation, match movement, camera tracking, rendering, video editing and compositing.

Blender also includes sculpting functions. Besides, it provides many tools and modifiers to simplify the creation of 3D printed meshes, including solutions to repair broken meshes. Blender is suitable for users that are not satisfied with the steep learning curve and are looking for one of the best free 3D modelling software, such as professional 3D modellers and 3D designers, game developers.

OpenSCAD is free software for creating solid 3D CAD models and can be used for multi-platforms [145]. Unlike most free software used to create 3D models, such as Blender, it does not focus on the artistic aspect of 3D modelling, but on the CAD aspect. Therefore, when users plan to create 3D models of machine parts, it may

be the application.

OpenSCAD is not an interactive modeller. Instead, it is like a 3D compiler, which reads the script file describing the object and renders the 3D model from the script file. It gives designers complete control of the modelling process. It can easily change any step in the modelling process or execute the design defined by configurable parameters.

OpenSCAD provides two main modelling techniques: the first is to construct solid geometry, and the second is to stretch 2D contours [146]. In addition to the 2D path used for stretching, design parameters can also be read from DXF files. In addition to DXF files, OpenSCAD can also read and create 3D models in STL and OFF file formats. [145]

Admittedly, OpenSCAD is not for everyone. For die-hard modelling enthusiasts, this is probably the best free 3D modelling software, and they are keen to see their code come to life in a 3D printer. It is suitable for hard mould CAD encoder.

In this thesis, we use OpenSCAD, because we can completely control the modelling process by configurable parameters once master it.

2.5.2 Image rendering

Pov-Ray and Mitsuba

As above said, we use PBR to render our 3D models. Different from usual CAD modelling software, the 3D rendering software may not only involve the creation of 3D models but also can produce realistically rendered visualisations.

Pov-Ray

The Persistence of Vision Ray Tracer, the most common abbreviation as Pov-Ray, is a cross-platform ray program that can design 3D models, render 3D scenes, and generate synthetic images from text-based scene descriptions [164]. The advanced features of POV-Ray are listed as follows:

1. A script-based scene description language (SDL) supporting macros and loops.

2. A pre-design library of ready-made scenes, textures and objects.

3. Support many geometric primitives and construct solid geometry. POV-Ray surpasses other rendering and modelling systems because it uses its mathematical definition to represent objects internally; all POV-Ray basic objects can be described with mathematical functions.

4. Several light sources, such as point light, parallel light, area light, shadowless light, *etc.*

5. Atmospheric effects, such as fog, smoke, and clouds.

6. Surface patterns for procedural textures and bump mappings, such as wrinkles, bumps, and ripples.

7. Support texture and rendering output in multiple image formats, including TGA, PNG and JPEG.

8. Extensive user documentation.

Pov-Ray can handle most situations, expect for simulating natural light, since it only has spot light source. However, in real life, the light comes from all directions, directly from the light source, or indirectly generated after rebounding from objects in the environment, and is partially absorbed in the process. In a sense, the entire environment around an object should be regarded as a light source. Hence, to advance our research, we use Mitsuba (specifically, a **light probe image** plugin of Mitsuba) to replace Pov-Ray for generating global illumination effect.

Mitsuba

Like Pov-Ray, Mitsuba [179] also comes with many different predefined modules to describe and render the scene: light sources, integrators, materials, objects, textures, camera models, *etc.* It also provides options for distributing rendering jobs across multiple computers and cores in a network or cluster to improve performance. As open-source and highly extensible software, existing modules can be customised, and new plugins can be easily implemented and deployed. Programmers can customise or extend the Mitsuba's module in three ways: C++ interface, Python interface, and XML scene format. Users can extend their documentation to create or change scenes,

set parameters and start rendering jobs. What is more, its highly customisable and powerful toolsets are the main reasons for choosing Mitsuba for light field acquisition and rendering simulation. In our experiments, we use the Python interface to control rendering jobs and scene files.

Light probe image

Illuminating computer-generated scenes with captured real-world light can improve realism and help integrate computer-generated and real-world images. Environmental maps and light probes are very important for image-based re-illumination and are used for many purposes. One of them is to capture the entire range of directions in a given environment to re-illuminate objects in the environment. They were firstly used by Debevec [180] in 1998.

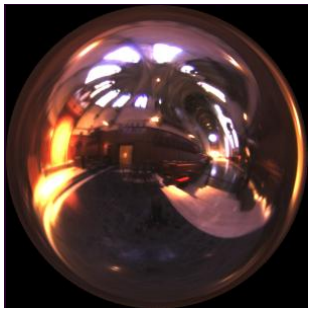
Light probes are created by taking one picture or two mirror sphere 90 degrees apart and assembling the two radiometric maps into a complete sphere. One photo is enough because when viewed from a given direction, the mirror ball can reflect the entire environment. However, part of the environment is masked by the camera's reflection in the mirror ball. Therefore, two mirror ball photos can be assembled at a 90-degree angle to create a photodetector. The area hidden by the camera in the first mirror ball appears in the second photo of the mirror ball. A very famous example is the environment map of the Cathedral of Grace (see Figure 2.12a).

The light probe image below is created from a single high dynamic range image of a mirror ball. Since only a single image is used, the image shows the camera and the photographer, and they are not well sampled in the area opposite to the camera (see Figure 2.12b).

2.6 Performance Metrics

There are various metrics which we can use to evaluate the performance of ML algorithms, classification as well as regression algorithms.Here, we are going to discuss various performance metrics that can be used to evaluate predictions for classification and regression problems.

Confusion Matrix



(a) Grace Cathedral. [180]

(b) Kitchen at 2213 Vine St. [180]

Figure 2.12: Light probe images.

It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes.Explanation of the terms associated with confusion matrix are as follows

- True positives (TP). It refers to the positive values of the correct prediction, that is, the values of the actual category and the predicted category are "yes".
- True Negatives (TN) - These are correctly predicted negative values, that is, whether the actual category and predicted category values are both "no".
- False positives (FP) - It indicates that the actual category is "no", but the predicted category is "yes".
- False negatives (FN) - A value indicates that the actual category is "yes", but the predicted category is "no".

Classification Report

This report consists of the scores of Precisions, Recall, F1 and Support. They are explained and used in our project as follows:

Generally, the retrieval of watermark bits can be seen as a binary classification problem with two symmetric classes. However, since in our case value 0 bits correspond to no alteration of the carrier surface and are indistinguishable from the background, we treat the retrieval process as a binary detection test. Thus, as performance measures of the overall watermark retrieval method, we report sensitivity (recall) TPR=TP/(TP+FN), specificity SPC=TN/(TN+FP), precision

PPV=TP/(TP+FP) and negative predictive value NPV=TN/(TN+FN), where TP and FN are the numbers of value 1 bits retrieved correctly and correspondingly incorrectly, TN and FP are the numbers of value 0 bits retrieved correctly and correspondingly incorrectly.

F1 score is an overall measure of the accuracy of the model that combines precision and recall. In other words, a high F1 score means that false-positive rate and the false-negative rate are both low, so we can correctly identify the actual threat without being disturbed by false positives. An F1 score of 1 is considered perfect, while a model of 0 is a complete failure. F1=2×TPR×SPC/(TPR+SPC).

AUC

AUC (Area Under Curve)-ROC (Receiver Operating Characteristic) is a performance metric, based on varying threshold values, for classification problems. As name suggests, ROC is a probability curve and AUC measure the separability. In simple words, AUC-ROC metric will tell us about the capability of model in distinguishing the classes. Higher the AUC, better the model. As I want to get direct insight about the accuracy of the retrieval information matrix, the classification report is mainly chose in my research project.

MAE

It is the simplest error metric used in regression problems. It is basically the sum of average of the absolute difference between the predicted and actual values. In simple words, with MAE (Mean Absolute Error), we can get an idea of how wrong the predictions were. MAE does not indicate the direction of the model i.e. no indication about underperformance or overperformance of the model. The following is the formula to calculate MAE=$\frac{1}{N}\sum_{i=1}^{N}|Y_i-\hat{Y}_i|$, Here, Y_i is Actual Output Values and $\hat{Y}_i$= Predicted Output Values.

MSE

MSE (Mean Squared Error) is like the MAE, but the only difference is that the it squares the difference of actual and predicted output values before summing them all instead of using the absolute value. The difference can be noticed in the following equation, MAE=$\frac{1}{N}\sum_{i=1}^{N}(Y_i-\hat{Y}_i)^2$, Here, Y_i is Actual Output Values and $\hat{Y}_i$= Predicted Output Values. The MSE is used in our neural network as loss

function, as it is great for ensuring that our trained model has no outlier predictions with huge errors.

Chapter 3

Watermarking of 3D Printed Objects

3.1 Introduction

In this chapter, we give a snapshot of the initial research plan that informed the research presented in this thesis. We also describe the results of an initial investigation on 3D watermark retrieval from 3D printed objects, which in later chapters serves as the baseline against which we compare more advanced deep learning based techniques.

3.2 3D mesh watermarking

With the advanced development of 3D technology and the rapid development of the Internet, copyright protection or content protection or authentication of 3D images or videos has become necessary and is now considered the most important field. 3D watermarking technology is a technology that hides certain information in a 3D model or 3D image or video in the spatial or frequency domain. This is one of the solutions for the above-mentioned purpose. Generally, 3D watermarking is a technology that embeds or hides information in audio, image or video for use in different applications (such as copyright protection, digital rights management (DRM), content authentication, etc.). Watermarking technology can be divided

into the following categories:

- Based on working domain: spatial and frequency.
- Based on the type of document to be embedded: text, images, audio, and videos.
- Based on Human perception: visible watermark, invisible robust watermark, invisible fragile and dual.
- According to application: Source-based and Destination based.
- According to the type of information needed at the time of extraction: blind and non-blind.

However, all these classifications and researches are based on digital 3D watermarking. There is no a clear classification for printed 3D watermarking. Hence, we proposed the printed 3D watermarking and our research directions.

3.3 Problem analysis

Since our project "Information embedding and retrieval in 3D printed objects" is dealing with a nascent research field, and the prior work on watermarking 3D printed objects, is particularly limited, there are no well-defined major research trends to follow. Therefore, in this chapter, we give a systematic overview of possible approaches to the problem and present results from initial explorations of some of the most promising research directions of this framework.

3.3.1 Proposed classification

The proposed classification of approaches to the problem of watermarking 3D printed objects is summarised as below. The first distinction we consider is that between visible and invisible watermarking. Visible watermarking embeds the watermark on the 3D printed object's surface or perhaps below that in cases where a transparent is used. Invisible watermarks are embedded under the surface of objects printed

on opaque materials. Below we describe some of the elements of that classification which are especially interesting and technically feasible to implement.

1. Visible Watermarking:

   (a) The watermark is **view independent**, that is, the same message is extracted from an image of the watermarked object, no matter the position and orientation of the camera relative to the object.

   - The message is embedded on the surface of the 3D printed object.
   - The message is embedded just below the surface of a semi-transparent object and is read from a back-lit photo.

   (b) The watermark is **view dependent**, that is, from different photos of the object, different information might be extracted, depending on the camera's position and orientation. This kind watermarking could be beneficial to some applications, since from the extracting watermark information we may infer information about the camera's location, or we can customise the embedded information depending on the camera's location.

   - Use a piece-wise flat surface as a carrier. The watermark information is embedded on more than one of the flat parts of the object.
   - Use a non-flat surface as a carrier. For example, the watermark is printed on a spherical, or a rippled surface, and different bits of information are extractable from different camera positions.

2. Invisible Watermarking, for example, ellipsoids embedded under the surface of an opaque object.

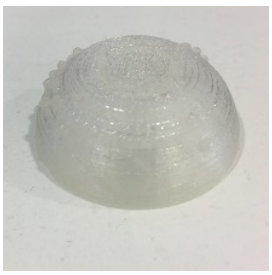
   - The ellipsoids are modelled as voids in the volumetric model of the 3D printed object.
   - The ellipsoids are printed with a material different than that of the main object.

Figure 3.1 shows proof-of-concept designs for some view independent members of the taxonomy. Depending on the type of the carrier surface, we embedded view

(a) Flat opaque.

(b) Non-flat opaque.

(c) Flat semi-transparent.

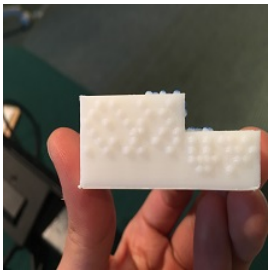


(d) Non-flat semi-transparent.

Figure 3.1: Proof-of-concept designs of view independent watermarks.

independent watermarks lying on a flat opaque surface (Figure 3.1a), a non-flat opaque surface (Figure 3.1b), a flat semi-transparent surface (Figure 3.1c), and a non-flat semi-transparent surface (Figure 3.1d). In all cases, we printed the 3D objects using a single material, and the watermark was embedded in the form of geometric texture, that is, semi-spherical or cuboid bumps.

Figure 3.2 shows proof-of-concept designs for view-dependent watermarks. In Figure 3.2a, in a rather trivial extension of view independent watermarking, different flat parts of an object, which are visible from different camera positions and orientations are used to carry different pieces of watermark information. In Figure 3.2b, a view-dependent watermark is embedded on a spherical surface printed with a yellow colour material. The sphere is split into four sectors using landmarks in the form of small bumps printed with a green colour material. These landmarks also carry a small amount of watermark information, while inside each spherical sector, more information is embedded in the form of bumps printed with a red colour material.



(a) Piece-wise flat surface.

(b) Non-flat surface.

Figure 3.2: Proof-of-concept designs for view dependent watermarks.

3.3.2 Identification of equipment requirements

The following equipment items and software tools were identified at the early stages of undertaking this research as potentially useful.

Hardware:

- 3D printers: Finder, Prusa i3 Mk2 multi-material, TAZ LULZBOT, Raise 3D N1. These were the models of 3D printers we had access.
- Filaments for 3D printing: PLA, HIPS, ABS, TPE, and Wood. Their properties are summarised in Table 3.1. In our in-house printing experiments, PLA was used most often because it is considered the most versatile and less demanding in terms of technical expertise 3D printer material.
- An iPhone 6s plus and a Huawei Mate 20 personal phones will be used as the main tools for taking photos of the 3D printed object.
- A Kern OBN 132 microscope was as an advanced tool to capture the fine texture of 3D printed models for pattern recognition. However, the large amount of noise of the 3D printed processes, made us abandon this research direction from an early stage.
- An XRadia/Zeiss VersaXRM 410, providing micron resolution x-ray tomography was used to take images of slices of 3D printed objects. However, the large overheads of the scanning process made us abandon this research direction too.

Software:

Table 3.1: Properties of various 3D printing materials.

Filament	Special properties	Strength	Flexibility
PLA	Ease of printing; minimal wrap	medium	low
HIPS	Dissolves in limonene; best support material for ABS	low	medium
ABS	Wear resistant; heat tolerant	medium	medium
TPE	High elasticity; impact resistant	low	high
WOOD	Similar to actual wood; printing behaviour similar to PLA	medium	medium

- OpenSCAD: academically oriented CAD software which provides a high precision and convenient modelling platform with advanced CAD features.
- Matlab: conveniently, it uses matrices as its fundamental data structure, and its graphical outputs are optimized for interaction.
- Pov-Ray and Mitsuba: these two 3D rendering software tools can produce realistically rendered images of 3D CAD models.
- PyCharm and Spyder: two Python IDEs which offer smart code assistance and provide useful tools that increase coding productivity.

3.4 On surface watermarks

In the rest of the chapter, we present some early results on selected members of the taxonomy. In all cases, the watermark retrieval algorithm will be based on the analysis of LBPs [49], a simple yet effective image processing and analysis method, which has been shown to work satisfactorily on various settings that are similar to ours, more details about LBP in section2.2.2. In particular, we extended Matlab's original LBP function of to a new function which can handle colour images and is computed faster because it uses a limited subset of 'uniform' patterns instead of the full set of rotation invariant patterns. We set two other parameters, "isRotInv" and "isChanWiseRot". The "isRotInv" is a logical flag, default false. When enabled generated rotation invariant LBP acquired via finding an angle at which the LBP or a given pixels minimal. Increases run time, and results in a relatively sparse

histogram. The "isChanWiseRot" is a logical flag, when enabled (default value) allows channel wise rotation. When disabled/false rotation carried out based on rotation of first color channel. Here, we all use default values.

View independent watermarks on flat surfaces

The retrieval algorithm combines LBPs with the Hough transform to support auto registration. All the photos were taken under natural light, and the 3D printed object should be near the photo's centre. The boundary of the rectangular watermarked area model should be relatively clear.

The flow of the algorithm is shown in Figure 3.3.

1. The input is an image of the 3D printed object; it can be a colour image.

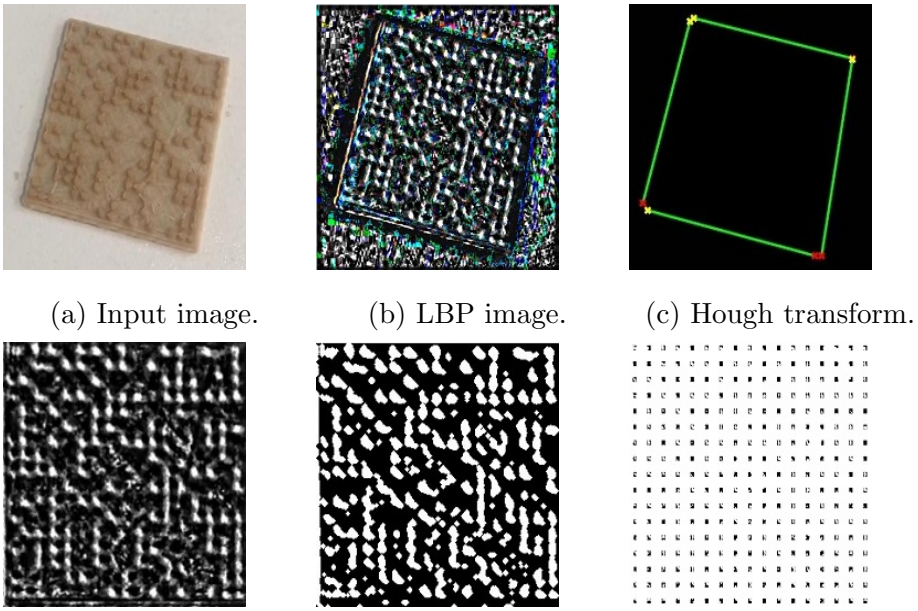
2. Process the input image by LBP. We use a modified LBP method, which is more robust against adverse lighting conditions. The LBP image is binarised using a global threshold.

3. Using Hough transforms, get the locations of the four corners of the watermarked area. The Hough transform based on line detection algorithm should detect the four lines forming the boundary of the watermarked area. From these four lines, compute the four corners of the watermarked area.

4. The auto registration step uses the locations of the four corners computed above to map the watermarked area into a square image. Typically, we map it into 256×256 image, with corners [0,0;255,0;0,255;255,255], such that the top left corner detected on the input image is mapped to [0, 0].

5. Split the registered image into smaller square cells arranged as a regular grid and from each cell retrieve one bit of the watermark. When the percentage of white to the black pixels is above 0.5, then the retrieved bit value is one. The 0.5 threshold is essentially a user variable, which can be adjusted accordingly.

6. Optionally, use ensemble learning [190] by averaging the retrieved watermarks from multiple images. If the ratio of ones at a particular element position in the retrieved bit arrays is above 0.8, then the bit value in the final ensemble



(a) Input image. (b) LBP image. (c) Hough transform.

(d) Auto registration. (e) Binarized image. (f) Cell grid.

Figure 3.3: The watermark retrieval pipeline.

bit array is set to one. Again, the 0.8 is a user variable which can be adjusted as required.

In Table 3.2, we show results from a limited experiment, based on six images of size 3024×4032 of different 3D printed objects. The images were taken under natural light, from different camera positions and orientations.

Table 3.2: Retrieval accuracy for view independent watermarks on planar opaque surfaces.

Item	Overall precision	Ones precision	Zeros precision
1	0.938	0.993	0.884
2	0.955	0.993	0.918
3	0.948	0.979	0.918
4	0.948	0.951	0.946
5	0.879	0.916	0.844
6	0.945	0.923	0.966

3.5 Under the surface watermarks

Besides retrieving 3D information from visible watermarks on 3D printed objects, we also did some preliminary work on invisible watermarks embedded under the object's surface.

Invisible watermarks embedded in a 3D printed object can only be seen by some special equipment, such as an X-ray scanner, or under special environmental illumination conditions. It is evident that, in some critical applications, such watermarks would have better security properties.

In a simple setting, for a proof-of-concept experiment which does not demand specialised hardware, we embed the watermarks just under the surface of thin objects printed on semi-opaque materials, and then the watermarks become visible under strong back-lighting.

3.5.1 Ellipsoidal watermarks

We chose to embed under the object's surface watermarks in the form of voids of ellipsoidal shape. In our experimentation, we did not define any specific watermark embedding and retrieval protocol. Instead, we experimented with various printing materials. We measured how accurately retrieved some key properties of the ellipsoidal voids are, whose properties can potentially be used to carry watermark information, such as the orientation angle of the ellipsoid's major axis, or the ellipsoid's aspect ratio.

3.5.2 Ellipsoid detection algorithm

The ellipsoid detection algorithms consist of two main steps. In the first step, we binarise the input image, and in the second, we detect ellipsoids by calling a standard Matlab function.

1. Binarisation: convert colour image to grey image; improve image contrast; convert grey image to binary image.

2. Ellipsoid detection: using Matlab's regionprops with default value to detect

Table 3.3: Estimated orientation angles of the detected ellipses.

Original			Pink			Brown			Dark Green			White		
0.0	10.0	20.0	1.0	10.9	19.2	0.0	14.0	23.3	1.0	8.6	18.9	-3.0	15.2	12.0
40.0	50.0	60.0	41.4	39.7	55.1	40.0	43.1	54.4	32.0	47.8	58.0	47.8	51.4	55.2
70.0	80.0	90.0	71.0	81.3	89.0	71.0	77.8	89.0	65.9	77.9	89.8	71.2	87.3	86.0
Average error			2.5			2.7			2.5			4.7		

elliptical objects and obtain their measurements, such as Orientation, MajorAxisLength, MinorAxisLength, Eccentricity. [191] More explain about regionprops in section 2.4.3.

Figure 3.4 shows some test models printed on different colour materials and the detected ellipses.

We designed two experiments for testing the robustness of the retrieval algorithm in extracting the orientation angle and the aspect ratio of the ellipsoids accurately, aiming at establishing whether these two characteristics of the ellipsoids are potentially suitable carriers of watermark information.

3.5.3 Orientation angle testing

The experimental protocol is as follows:

1. Design a thin square model with nine ellipsoids embedded inside it, each of the ellipsoids being oriented at a different angle between its major axis and the base of the square.

2. 3D print the thin square using four different colours, see Figure:3.5.

3. Compute the rotation angle of each detected ellipse and compare it against the original angle.

From the retrieved orientation angles shown in Table 3.3, it can be seen that the highest error appears on the White object, which is the most transparent and as a result the of the back-lit image is no sharp, challenging the robustness of the ellipse detection algorithm. It is a somewhat counter-intuitive result since one would expect the more transparent material to give the best results.

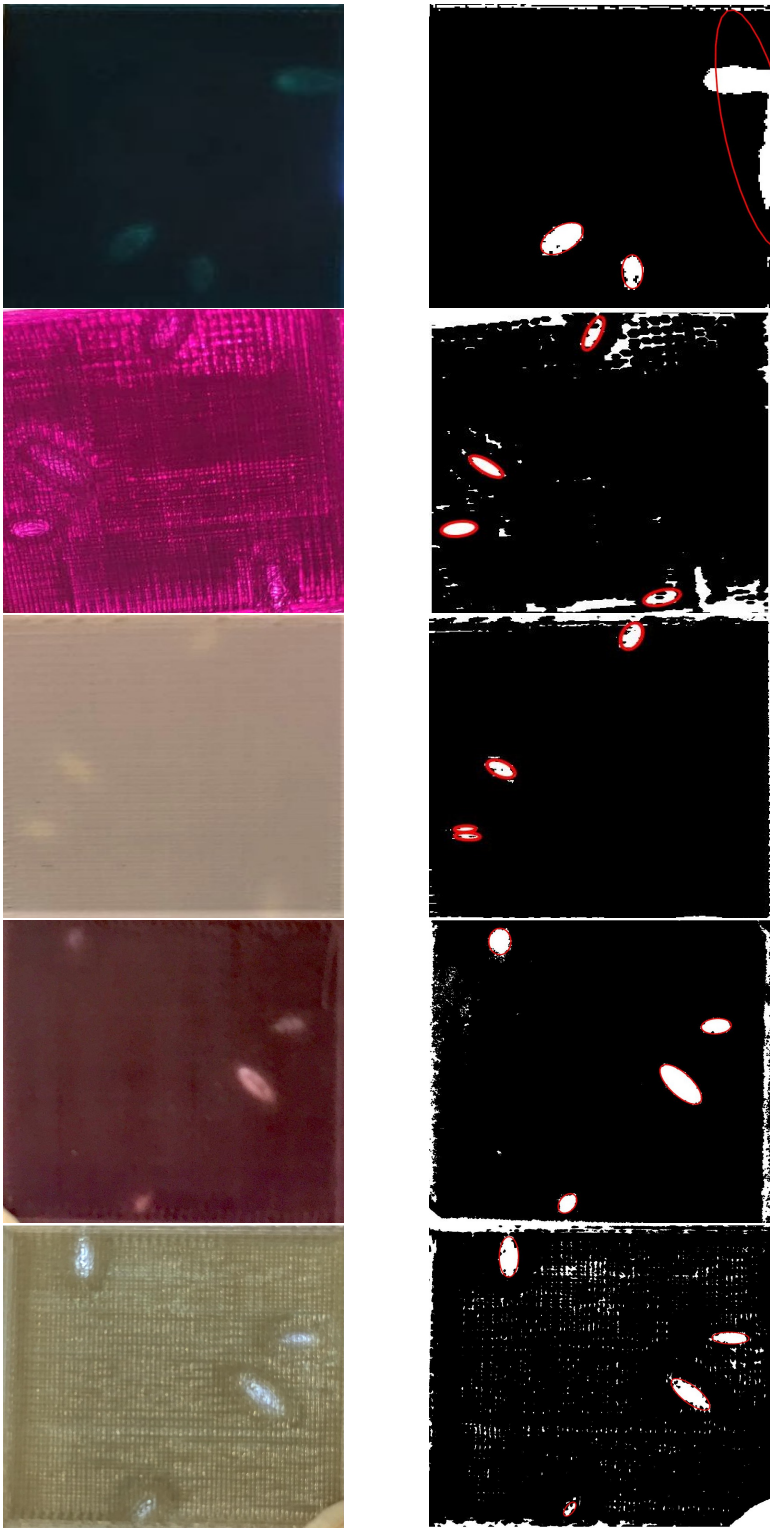
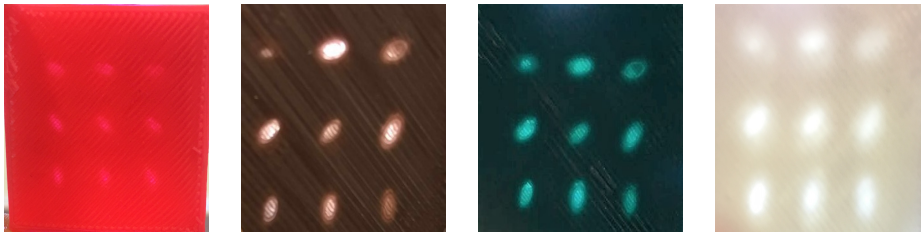


Figure 3.4: **Left:** Watermarked printed objects. **Right:** Detected ellipsoids



(a) Pink (b) Brown (c) Dark Green (d) White

Figure 3.5: A thin square with 9 ellipsoidal voids inside it, printed on 4 different materials.

We also notice that higher errors appear on angles around 45. The distributions of the error seem like following a normal distribution centred at 45. please refer Table 3.3 However, the validity of this inference is limited by the small number of data, and it should be investigated further.

3.5.4 Aspect ratio testing

The experimental protocol is as follows:

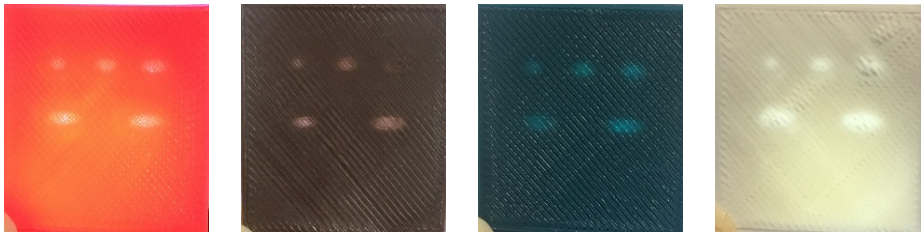
1. Design a thin square model with five flat ellipsoids embedded inside it, each of the ellipsoids having a different ratio between the first and the second axis lengths.

2. 3D print the thin square using four different colours, see Figure 3.6.

3. Compute the aspect ratio of each detected ellipse and compare it against the original aspect ratio.

Table 3.4: Estimated orientation angles of the detected ellipses.

	1.5	2.0	2.5	3.0	3.5	Average error
Pink	1.2(-0.3)	1.8(-0.2)	2.0(-0.5)	2.4(-0.6)	2.9(-0.6)	0.44
Brown	1.3(-0.2)	1.7(-0.3)	1.3(-1.2)	2.3(-0.7)	2.9(-0.6)	0.60
Dark green	1.3(-0.2)	1.5(-0.5)	1.9(-0.6)	2.0(-1.0)	1.8(-1.7)	0.80
White	1.7(+0.2)	1.3(-0.7)	1.7(-0.8)	1.7(-1.3)	1.8(-1.7)	0.94
Average error	1.4(-0.2)	1.6(-0.4)	1.7(-0.8)	2.1(-0.9)	2.4(-1.1)	



(a) Pink (b) Brown (c) Dark Green (d) White

Figure 3.6: A thin square with 9 ellipsoidal voids inside it, printed on 4 different materials.

Regarding the aspect ratio, it can be seen from Table 3.4 that, as expected, the smaller the ratio is, the higher the precision of its estimate. It can also be seen that the aspect ratio precision is affected by the transparency of the 3D printing material in the same way as that of the orientation angle, that is, when the material is less transparent, the precision is higher.

From the above results, we can conclude that retrieval of 3D watermarking information from back-lit, thin objects, 3D printed on semi-opaque material is feasible. However, the optimisation of embedding and retrieval algorithms seems to be a challenging task, as there is a multitude of variables, and the behaviour of some of them is unpredictable.

3.6 Conclusion

This chapter contains a snapshot of my initial research plan and describes various possible research directions about designing and retrieving 3D watermarking from printed 3D objects. In this chapter, the traditional method, LBP, seems useful for extracting these 3D watermark bumps, but it requires favourable conditions, in particular, light, camera angle, image quality. As the preliminary exploration, it is clear that the method will be difficult to adapt and apply to various real application scenarios, but it is a good benchmark for following research works. In the following chapters, DNN is employed as the primary tool to be used in the project.

Chapter 4

Single Image Watermark Retrieval from 3D Printed Surfaces via Convolutional Neural Networks

In this chapter we propose and analyse a method for watermarking 3D printed objects based on a Convolutional Neural Network, instead of the Local Binary Patterns used in Chapter 3, aiming at increasing the overall retrieval accuracy by increasing the generalisation power of the algorithm into objects printed on previous unseen materials. In this chapter, the watermarks are on a planar region of the 3D printed object and have the form of small semi-spherical or cubic bumps, arranged at the nodes of a regular grid. Experiments verify that in the retrieval algorithm has indeed higher accuracy rate than the LBP based one.

4.1 Introduction

Regarding more traditional image processing the analysis techniques, in an earlier approach, we developed a technique based on LBP [38] in the last chapter to recognise the watermark bumps features. However, as it is often the case with handcrafted feature extraction methods, the results did not generalise very well under adverse conditions, in particular, when the background patterns were too prominent, or under extreme uneven illumination, or unfavourable camera viewpoints.

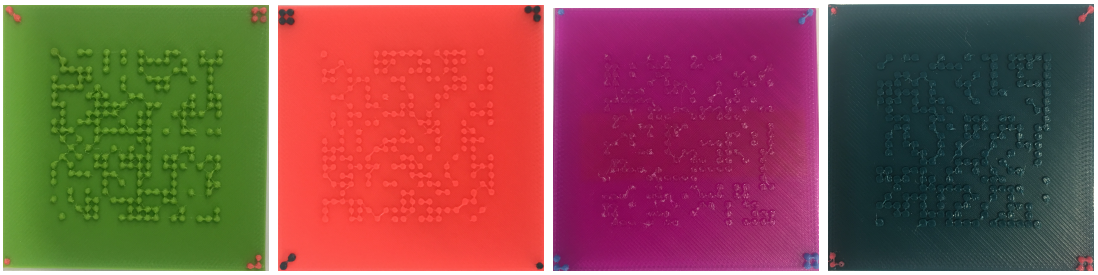


Figure 4.1: Examples of 3D printed surface watermarks.

The watermark is similar to DotCode. The information is embedded on the nodes of a regular grid, one bit per node, using semi-spherical or cubic bumps. The existence of a bump corresponds one value at the location of the grid and its absence to zero value. Here, we restrict ourselves to planar surfaces and leave the generalisation of the technique to other surfaces as future work. See Figure 4.1 for an example.

The proposed watermarking of 3D printed surfaces has some apparent similarities with digital 3D watermarking. Indeed, the construction of a 3D printed object starts with the creation of a digital file, which in the case of watermarked 3D printed objects is a watermarked 3D file, and which up to 3D printer limitations and 3D printing uncertainties it specifies the printed object completely. However, the two processes could diverge significantly during the retrieval process. In the case of digital 3D watermarking, the input of the retrieval algorithm is a digital file, and an inversion of the embedding process retrieves the watermark. The challenge, in this case, is to make the embedding-retrieval cycle robust against a variety of malicious and unintentional attacks which could even include 3D printing and re-scanning [147, 148]. On the other hand, in our approach of 3D printed watermark retrieval, there is no need to reconstruct the 3D file. Instead, we treat watermark retrieval as a computer vision problem, aiming at retrieving the watermark from a single photo of the printed object, with no need for using any specialised equipment such as laser scanners.

We investigate the extent to which deep neural networks can overcome the multitude of challenges that a realistic scenario would pose into the retrieval of such watermarks. In particular, we assume:

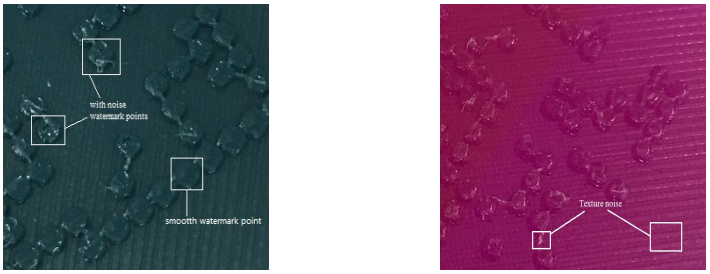


Figure 4.2: Left: poor printed watermarks bumps. Right: background artifacts.

Figure 4.3: Watermark retrieval challenges due to illumination. Left: natural light. Right: uneven artificial light.

1. Use of a single material for the main watermark, meaning that the carrier surface and the bumps on it are of the same colour.

2. Watermark retrieval from a single image from a hand-held phone camera.

3. Use of a low end printer with plastic filaments, meaning that the carrier surface may contain significant noise as well as various infill patterns created during slicing. For that reason, the sizes and shapes of the bumps may vary considerably, even if they are identical in the digital file. See Figure 4.2.

4. Arbitrary viewpoint and variability of lighting conditions, from natural light to uneven artificial illumination. See Figure 4.3.

The main contribution of this chapter is to design a CNN (we named CNN-3DW) to effectively retrieve 3D watermark bumps. It takes an RGB image as input and outputs a density map representing the probability of a pixel to be on a watermark bump. The next steps are straightforward, that is, the registration of density map image, followed by rather a simple image processing and analysis operations. In a limitation of our approach, we opted for a convenient solution into what is a classic

image registration problem, aiming at decoupling the primary machine learning component from the subsequent image processing and analysis steps. We used the second material of distinct colour to mark the four corners of a rectangular region of interest. As a robust convenience solution to the watermark orientation problem, we used four distinct corner symbols consisting of one, two, three and four small dots, respectively.

4.2 Method

In all of our experiments, the watermark was a 20×20 binary matrix. Using filaments of nine different colours, we 3D printed 16 objects in total, each one carrying a different, randomly generated watermark.

4.2.1 Dataset

From each printed object, we captured 15 images of size 3024×4032 under different, arbitrary perspectives, ten of them under natural light and five under extreme artificial illumination. In total, we captured 240 images, 80% of which were used for training and 20% for validation. As we did not have sufficient training images to train the proposed CNN, we used data augmentation methods to deal with a small training set problem. On each training image, we applied three random rotations followed by a random change of its average brightness level. We then randomly sampled ten patches from each rotated image, creating a training set of 5760 images of size 512×512.

The original 192 training images of size 3024×4032 were annotated by hand, creating binary images which had value one at the centres of the watermark bumps and value 0 elsewhere. These binary images were downsampled to size 756×1008 to fit the desired size of the CNN-3DW output. The downsampled binary images were converted to smoother density maps by applying on them one pass of a Gaussian filter with $\sigma = 5$. The filtering process also increased the support of the constructed density maps with the regions of non-zero values roughly corresponding to the watermark bumps, see Figure 4.4.

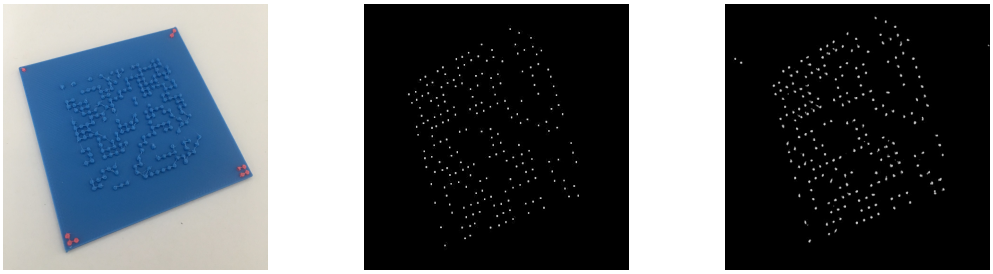


Figure 4.4: Left: test image. Middle: ground truth density map obtained by hand annotating the original the image. Right: the density map estimated by CNN-3DW.

4.2.2 CNN-3DW architecture

Zhang *et al.* [184] proposed a simple but effective Multi-column Convolutional Neural Network (MCNN) architecture to map the image to its crowd density map. The features learned by each column CNN are adaptive to variations in people/head size due to perspective effect or image resolution. Here, the CNN-3DW is based on MCNN. As the various camera views introduce different types of perspective distortion, the size of watermark bumps may differ from image to image. However, since the distance between the camera and the printed object is large relative to the size of the object, all bumps in one image are of the same scale size. For that reason, we opted for a one-column deep neural network to learn the relationship between the original images and the constructed density maps. According the experiments, the network consists of five convolutional layers with kernel sizes 9×9,7×7,7×7,7×7,1×1 and feature map channels 48, 96, 48, 24, 1, respectively.The reason we designed our network based on FCN rather than a deeper architecture such as VGG and ResNet is that we found in our experiments that the output of FCN was more stable. Indeed, locating dots on a single coloured plate is a relatively simple task and the adoption of deeper models is more resource expensive.

The overall structure of the CNN-3DW is shown in Figure 4.5. There are two layers of max-pooling applied on 2×2 regions, and the activation function is the Rectified Linear unit (ReLU), which generally performs very well in CNNs, see for example [149]. At the top of the CNN-3DW, instead of a fully connected layer, we use a convolutional layer of kernel size 1×1. The number of layers and the filters sizes in each layer, have been chosen through extensive experimentation and

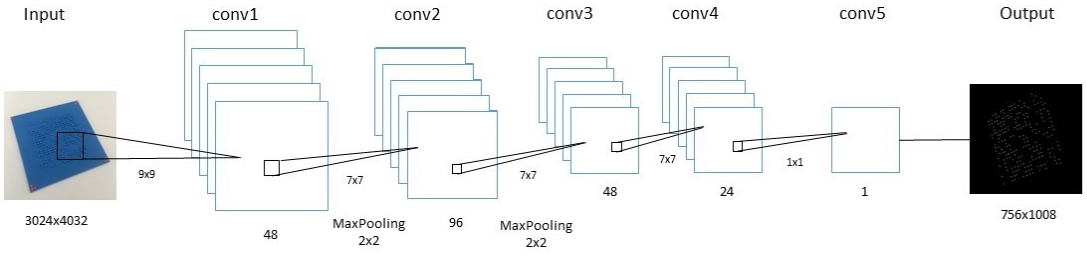


Figure 4.5: The architecture of the proposed CNN-3DW.

cross-validation.

To train the network, we use the mean squared error (MSE) loss function:

$$L(\Theta) = \frac{1}{N \times H \times W} \sum_{n,h,w} (f(\mathbf{X}; \Theta) - \mathbf{Y})^2, \tag{4.2.1}$$

where Θ is the set of network parameters; N is the number of training images; $\mathbf{X}$ is an input image, $\mathbf{Y}_i$ the ground truth confidence map of $\mathbf{X}$ and $f(\mathbf{X}, \Theta))$ is the predicted confidence map of the input image X; W and H are the width and height of the confidence map image $\mathbf{Y}$ respectively.

We trained the CNN-3DW with Keras built on TensorFlow, using an NVIDIA GeForce GTX TITAN X. We used Adam optimization with a learning rate of $1e-5$ and $1e-6$ after 50 epochs, batch size 2 and momentum 0.9. The average training time was about seven hours.

4.2.3 Image registration and matrix retrieval

All models carry four landmarks printed with a different colour and can be easily located using colour segmentation and then used as control points to compute an affine transformation for the image registration. The output density map of the CNN-3DW is thresholded; regularised by applying the same affine transformation on it, and Matlab's *regionprops* function is called to detect its connected regions and obtain estimates of their centroids and their two semi-axes. If the sum of the two semi-axes is above a threshold, we classify that region as a watermark point located at the centroid of that region. See Figure 4.6.

Finally, to transform the potentially noisy pixel coordinates of the centroids into

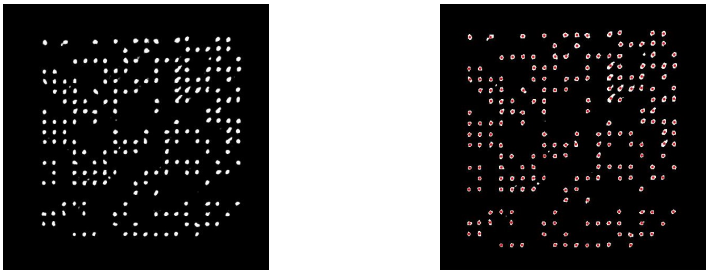


Figure 4.6: Left: the regularised CNN-3DW output. Right: the detected centroids of watermark bump regions.

row and column indices of the watermark matrix we apply k-mean clustering on the x and y coordinates separately, and obtain row and column indices as the indices of the two clusters a centroid belongs, respectively.

4.3 Experimental results

The test set consisted of three randomly selected images of each printed object, two of them under natural light and one under extreme artificial illumination, for a total of $3 \times 16 = 48$ images.

While during the development of the CNN-3DW network we used Mean Absolute Error and Mean Square Error to quantify the accuracy of the estimated density maps, here we focus on the evaluation of the overall watermark retrieval method and report sensitivity (TPR), specificity (SPC), precision (PPV) and negative predictive value (NPV). The results are summarised in Table 4.1.

From Table 4.1 we notice that, generally, the method works well and in conjunction with the use of error correction codes seems to be a viable solution for embedding machine readable information in a physical object. Regarding material suitability, we notice that the results are slightly worse when dark colours are used, such as Dark Brown or Dark Green. We also notice that the algorithm performs worst with the Transparent Purple material, something that could be attributed either to the transparency of the material or to the use of one only 3D printed object from that material, while most of the other materials were used to print two objects.

For the challenges shown in Figure 4.2 and 4.3 in particular, our method performs

Table 4.1: Evaluation of the retrieval method in different printed colours.

	Colour	TPR	SPC	PPV	NPV
1	Green	0.965	0.966	0.967	0.965
2	Dark Brown	0.798	0.863	0.854	0.812
3	Blue	0.964	0.998	0.998	0.966
4	Dark Green	0.662	0.698	0.699	0.661
5	Wooden	0.830	0.840	0.851	0.818
6	Bronze	0.889	0.901	0.900	0.890
7	Luminous Red	0.971	0.981	0.982	0.970
8	Skin	1.000	1.000	1.000	1.000
9	Green	0.994	1.000	1.000	0.993
10	Dark Brown	0.905	0.940	0.937	0.912
11	Blue	1.000	0.998	0.998	1.000
12	Dark Green	0.882	0.942	0.939	0.887
13	Wooden	0.862	0.881	0.871	0.872
14	Trans Purple	0.542	0.862	0.756	0.685
15	Luminous Red	1.000	1.000	1.000	1.000
16	Skin	0.882	0.921	0.914	0.893
	Average	**0.895**	**0.897**	**0.895**	**0.885**

relatively well. For the objects in Figure 4.2 (left) image, the TPR arrives 0.961, while for Figure 4.2 (right), the TPR is 0.860. Under the natural light illumination shown in Figure 4.3 (left), its retrieval accuracy is 0.980, while under the artificial uneven lighting in Figure 4.3 (right), the accuracy drop to 0.812.

4.4 Conclusion

It presented a method for watermarking 3D printed objects, focusing on watermark retrieval which, in contrast to a digital 3D watermarking case, here is a very challenging computer vision problem. Our results show that CNN based computer vision techniques are capable of tackling effectively challenges related to the variability of parameters such as camera view, illumination conditions and printing material.

In future, we will try to extend our methods to 3D printed watermarks on non-developable surfaces [185], addressing this way a limitation of a common practice for embedding machine-readable information on an object, that is, printing on paper a DotCode and sticking it on the object's surface.

Chapter 5

Watermark Retrieval from 3D Printed Objects via Convolutional Neural Networks

In this chapter, we build on the material presented in Chapters 4, proposing various algorithmic improvements to the CNN based retrieval algorithm and conducting more extensive validation experiments. A majority vote by more than one images of the 3D object is employed, a technique compatible with our target application scenario since modern mobile phones capture multi-photos by a single shot. Moreover, an active learning method is proposed, in which we add to the training set examples on which the current network underperforms, and we retrain.

5.1 Introduction

Currently, a printed DotCode code usually only can be placed on flat surfaces of physical objects and very easily be vulnerable to various malicious attacks. For example, an attacker could replace a printed DotCode with their own, or overlay it. The root cause of these vulnerabilities is that the objects manufacturing and additional information embedding on these objects are two separate processes at the moment. For example, when a bottle shampoo is produced, its product details and batchcode are printed later, so this information is easily removed or falsified.

However, 3D printing brings a unique opportunity to address the problem by integrating the information embedding process into the manufacturing process. That is, the embedded information can be part of the fabric of a 3D printed object, rather than been manufactured separately and glued on to it.

Challenges

The challenges in developing a 3D printing watermark method can be summarized as follows:

1. Since 3D printed objects are often created from a single material, the background surface and the watermark carrying bumps would often have the same colour, and it would be difficult to distinguish between them.

2. The popularity of low-end printers with plastic filaments means that the carrier surface would often contain significant noise as well as various infill patterns created during slicing. For that reason, the sizes and shapes of the bumps may vary considerably, even if they are identical in the digital file. See Figure 5.1 (left).

3. Watermark retrieval is expected to be done from images captured by a handheld phone camera under potentially adverse conditions. Apart from assuming an arbitrary camera viewpoint, the variability of the lighting conditions, which may range from smooth, natural light to extremely uneven artificial illumination, should also be taken into account. See Figure 5.1 (middle and right).

Our experimental results show that convolutional neural networks can overcome this multitude of watermark retrieval challenges entailed into a realistic scenario.

Contributions

The main contributions of this chapter are the development of a watermark retrieval algorithm, the design and creation of an image database of 3D printed watermarked objects. The retrieval algorithm starts with a convolutional neural network generating an approximate density map representing the locations of watermark bumps

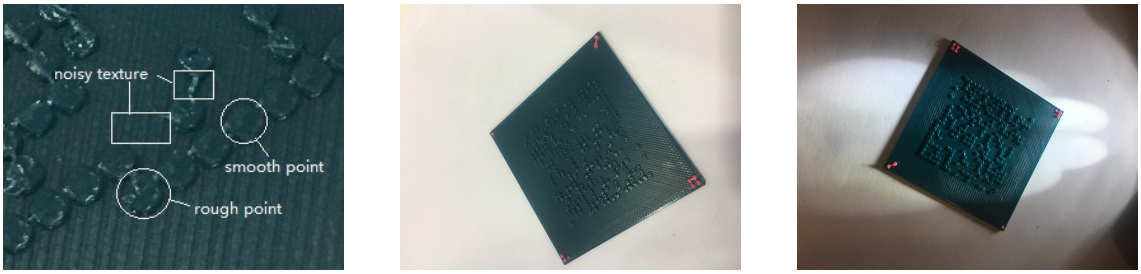


Figure 5.1: **Left:** Challenges related to the printing process. **Middle and Right:** Watermark retrieval challenges due to illumination: natural light (middle) and uneven artificial light (right).

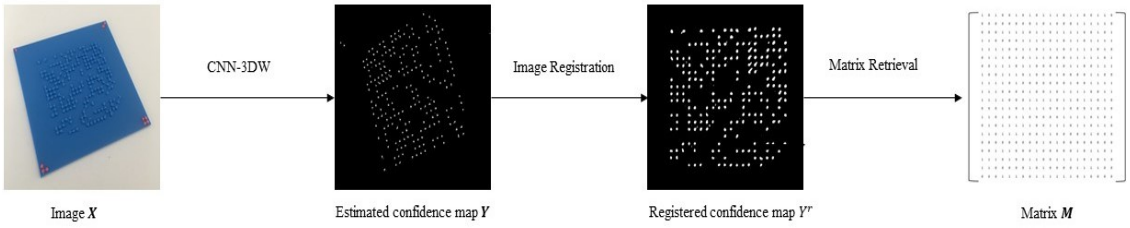


Figure 5.2: The pipeline of the proposed 3D watermarking retrieval method.

in the input RGB image. The subsequent steps are a series of simple image processing and statistical operations, specifically, the registration of the density map, a sequence of simple image processing operations, and finally, the retrieval of the watermark through the K-means clustering method. See Figure 5.2 for a high-level description of the algorithm.

5.2 Method

Given an image/photo of a printed watermark, as shown in Figure 5.1, we aim to extract the information embedded in the watermark, which in our case is a binary matrix. To detect the surface bumps corresponding to bit value one, we propose an image processing pipeline comprising of three stages (c.f. Figure 5.2). In the first stage, a fully convolutional network (FCN) takes an image $\mathbf{X}$ as input and outputs an estimated confidence map $\mathbf{Y} = f(\mathbf{X})$. To overcome the issues of target scaling, rotation and affine distortion in the input images, in the second stage, image registration procedures are applied to the confidence map $\mathbf{Y}$ to get a registered con-

fidence map $\mathbf{Y}^r$. Finally, the information matrix $\mathbf{M}$ is extracted from the registered confidence map $\mathbf{Y}^r$. In the following subsections, we explain these three stages in detail.

5.2.1 CNN-3DW

Network Architecture

The proposed CNN based model for 3D watermark retrieval is a fully convolutional network named **CNN-3DW**. The architecture is shown in Figure 5.3. It consists of five convolutional layers in which there are 48, 96, 48, 24, 1 kernels with sizes of $9 \times 9, 7 \times 7, 7 \times 7, 7 \times 7, 7 \times 7$ and 1×1 respectively. The activation function is the Rectified Linear unit (ReLu), which generally performs well in CNNs, see for example [149]. Two max pooling layers with step size two are added after the first and second convolutional layers, respectively. For all the convolutional layers, we set the step size as one and do zero paddings so that the image size is not altered by the convolution operations. As a result, the final output has one fourth the size of the input image due to two max pooling layers. Since the whole network is fully convolutional, it is able to handle input images of different sizes.

Ground Truth Confidence Map

Given an input image $\mathbf{X}$ of a printed watermark, the goal of our CNN-3DW is to estimate the confidence map of "bump" locations representing the value of ones in the information matrix. Figure 5.4 shows an image of a watermarked object and its corresponding ground truth confidence map and the estimated confidence map by the proposed CNN model. We need to annotate the training data for such purpose. Specifically, we generate the groundtruth confidence map $\mathbf{Y}$ based on the annotation image $\mathbf{A}$ by a Gaussian smoothing process.

$$\mathbf{Y} = \mathbf{A} * \mathbf{G}_\sigma, \tag{5.2.1}$$

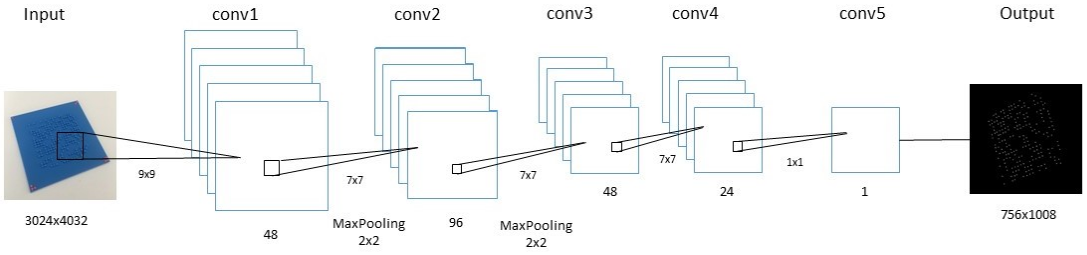


Figure 5.3: The architecture of the proposed CNN-3DW.

where $*$ is a convolution operator applied on the binary annotation image $\mathbf{A}$ and the Gaussian kernel $\mathbf{G}_\sigma$. The annotation image can be represented by

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is the centroid of a marked region,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.2.2}$$

The annotation images are the results of human annotation on the training data while the standard deviation of Gaussian kernel was determined empirically to be $\sigma = 5$.

Model Training

To train the network, we use the mean squared error (MSE) loss function:

$$L(\Theta) = \frac{1}{N \times H \times W} \sum_{n,h,w} (f(\mathbf{X};\Theta) - \mathbf{Y})^2, \tag{5.2.3}$$

where Θ is the set of network parameters; N is the number of training images; $\mathbf{X}$ is an input image, $\mathbf{Y}_i$ the ground truth confidence map of $\mathbf{X}$ and $f(\mathbf{X},\Theta))$ is the predicted confidence map of the input image X; W and H are the width and height of the confidence map image $\mathbf{Y}$ respectively.

Data augmentation techniques have proved to be beneficial to the training of deep CNN models, alleviating the overfitting issue, especially in situations where the training data is not large enough. In our experiments, we employ three data augmentation techniques to compensate for insufficient training data. Firstly, the images are randomly rotated by a random angle degrees. Secondly, the training data are augmented by randomly changing their average brightness. Finally, random

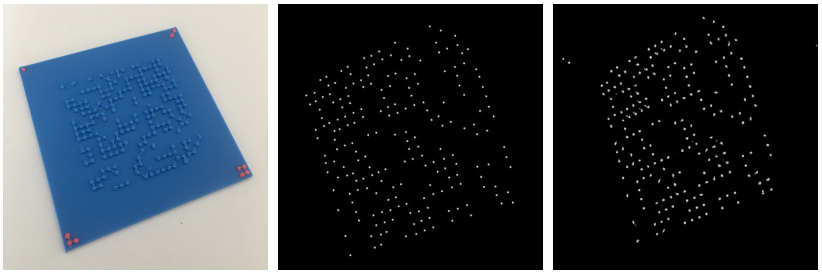


Figure 5.4: **Left:** image of a watermarked object. **Middle:** ground truth map.**Right:** corresponding confidence map.

cropping is applied to generate ten patches of uniform size (i.e. 512×512) from each image.

We trained the CNN-3DW with Keras [152] built on TensorFlow. The Adam optimizer [150], a stochastic gradient descent optimization method for training deep learning models, is employed with the default parameter values. We set the initial learning rate as $1e-5$ and decrease it to $1e-6$ after 50 epochs. The training is stopped after 100 epochs, where we observed convergence of the model. When using an NVIDIA GeForce GTX TITAN X, the average training time was about 7 hours on our own dataset which will be introduced in the following section.

5.2.2 Image registration, processing and matrix retrieval

Using the trained CNN-3DW model, we are able to estimate a confidence map $\mathbf{Y}$ of the watermark bumps in the image $\mathbf{X}$. To extract the embedded information matrix from the estimated confidence map $\mathbf{Y}$ is still non-trivial. The first issue is that we need to locate the watermark regions in the confidence map. In our experiments, we observed that noises exist in the background regions of some images, posing great challenges to the automatic watermark region localization. Significant bias in the watermark region localization would lead to catastrophic error in the following stages. To fight this issue off, we print four *landmarks* at the corners of the watermark with easily distinguishable colours (see Figure 4.1). During retrieval, the watermark regions are easily located by finding the "differently coloured" *landmarks* at the four corners.

The second issue is that the watermark in the image could be of variant scales,

rotations and affine distortions. To tackle this issue, we use image registration [165] to transform the watermark region in the confidence map $\mathbf{Y}$ to a square region (see Figure 5.2). In image registration, the *landmarks* are used as control points which are supposed to be mapped to the four corners of the new square region. In our experiments, we used Matlab's built-in tools for image registration [192]. After image registration, we obtain the transformed confidence map and the watermark region is cropped with the background region discarded. We denote the reserved watermark region of confidence map as $\mathbf{Y}^r$.

Finally, we extract the embedded information matrix $\mathbf{M}$ from $\mathbf{Y}^r$. The registered confidence map $\mathbf{Y}^r$ is firstly binarised by a threshold t, The binary confidence map $\hat{\mathbf{Y}}^r$ is visualized in Figure 5.5. The point falling into the i-th cluster will be assigned to a new coordinate in the $m \times m$ matrix $\mathbf{M}$.

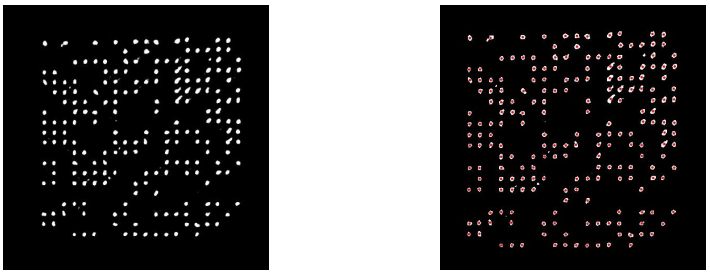


Figure 5.5: Left: the regularised CNN-3DW output. Right: the detected centroids of bit value 1 regions.

5.3 Experiments

In this section, we conduct a series of experiments to evaluate the effectiveness of the proposed approach to 3D watermark retrieval. Since there are no datasets with images of 3D printed objects watermarked with a similar method, we designed our dataset named **3DW-FS** [1]. We conduct five experiments and report the results in the following subsections.

[1]The whole dataset including raw images and the annotations will be made publicly available.

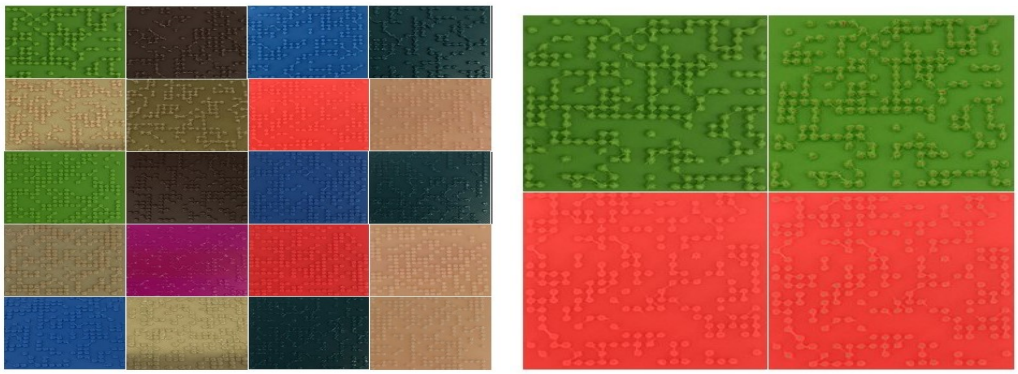


Figure 5.6: **Left**:exemplar images of objects 1-20 from left to right and top to down; **Right**: two example of raw images (left) and annotated images (right) for ground truth bump location extraction.

5.3.1 Dataset: 3DW-FS

To collect a dataset as the test bed for our validation experiments, we first print watermarks on multiple objects, then collect images of the watermarked objects for our experiments. The watermarks printed on objects convey information encoded in 20×20 binary matrices which are randomly generated and different for each object. The images are captured using a mobile phone camera which takes images of 3024×4032 resolution. The dataset consists of 290 images from 20 objects printed with nine different materials of different colours. We collect 15 images for each of the objects 1-16 and 5 images for each of the objects 17-20.

To simulate potential scenarios in practical applications, we captured images under variant conditions. Firstly, we collected 10 and 5 images under natural and extreme artificial illumination respectively for objects 1-16; while for objects 17-20, we collected 3 and 2 images under these two illuminations conditional respectively. Secondly, we keep various distances to the objects when collecting images, so that the watermark regions in the images have different sizes. Thirdly, we ensure that the watermarks in the collected images have various rotations in both horizontal and vertical directions to expect that the watermark can be recognised from arbitrary perspectives. As a result, our dataset is challenging and provides a simulation of real scenarios.

For CNN model training, we manually annotate all the images of objects 1-

16, i.e., 240 images in total. For the annotation, we put a dot at the centre of a watermark bump in the image, as shown in Figure 5.6 from which the ground truth bump locations can be easily extracted. The ground truth targets of the CNN model are then computed based on the annotated bump locations by Eq.(5.2.1).

5.3.2 Experimental Results

We report results from four tests.

1. A 5-fold cross-validation test on the 16 printed objects that were used for the development of the CNN-3DW.

2. A validation test on images from four objects printed after the development of the CNN-3DW was complete.

3. A cross-material validation test where training and test sets do not contain images of objects printed with the same material colour.

4. A test demonstrating the feasibility of an active learning approach [186], in which we identify a material colour on which the algorithm performs poorly, print more objects using that material and use their images to retrained the network.

Experiment 1

In this experiment, we assess the overall accuracy of the method and how it is affected by material colour and illumination conditions. We use objects 1-16 which are printed using nine material colours in total. From each object, we capture 15 images of size 3024×4032 with an iPhone 8 camera under different, arbitrary perspectives, 10 of them under natural light and 5 under extreme artificial illumination. From each object, we select two natural and one artificial illumination image to create a test set containing 48 images in total, while the other 192 images used for training. We repeat this process five times until all images have been used once for testing.

The results are summarised in Table 5.1. We can see that the proposed method performs well on most of the printed objects, dark green and transparent purple

being the most notable exceptions. Notice that in this experiment we used only one transparent purple object. Thus the poor material properties and not enough training data could explain the poor retrieval performance. In experiment 4, we employ a form of active learning to study this further.

Table 5.2 shows average performances overall 16 printed objects for each illumination condition. As expected, extreme artificial light has a negative effect on performance.

Experiment 2

For this experiment, we use images of objects 1-16 for training and test with images of objects 17-20. Notice that while in the previous experiment the watermark matrices of the test images also appear in the training set, with this test we can exclude the remote chance that CNN-3DW is memorising parts of watermark matrices.

The results are shown in Table 5.3. Apart from averages, we also report the results of an ensemble algorithm determining the value of each bit by a majority vote over the five images. Notice that the ensemble method is very much within the spirit of our target application scenario since modern mobile phones can easily capture and process short video sequences. The results show that CNN-3DW retrieval rates are independent if the watermark matrix and that majority vote can improve retrieval accuracy to levels that would be deemed satisfactory in a real-life application.

Experiment 3

Here we use the same test set as in Experiment 2, but each of the four newly printed test objects is tested on a neural network trained only with images of different material colours. This test aims to assess the generalisation ability of CNN-3DW on images from objects with unseen material colours.

The results are shown in Table 5.4. From the comparison between the results of Tables 5.3 and 5.4, we notice that CNN-3DW can handle some colours such as Blue and Dark Green quite well as unseen colours, while in some other cases (such as Wooden and Skin) the performance drops significantly.

Table 5.1: Experiment 1: 5-fold cross validation.

	Colour	TPR	SPC	PPV	NPV
1	Green (a)	0.965	0.966	0.967	0.965
2	Green (b)	0.994	1.000	1.000	0.993
3	Dark Brown (a)	0.798	0.863	0.854	0.812
4	Dark Brown (b)	0.905	0.940	0.937	0.912
5	Blue (a)	0.964	0.998	0.998	0.966
6	Blue (b)	1.000	0.998	0.998	1.000
7	Dark Green (a)	0.662	0.698	0.699	0.661
8	Dark Green (b)	0.882	0.942	0.939	0.887
9	Wooden (a)	0.830	0.840	0.851	0.818
10	Wooden (b)	0.862	0.881	0.871	0.872
11	Luminous Red (a)	0.971	0.981	0.982	0.970
12	Luminous Red (b)	1.000	1.000	1.000	1.000
13	Skin (a)	1.000	1.000	1.000	1.000
14	Skin (b)	0.882	0.921	0.914	0.893
15	Bronze	0.889	0.901	0.900	0.890
16	Transparent Purple	0.542	0.862	0.756	0.685
		0.895	**0.897**	**0.895**	**0.885**

Table 5.2: Performance comparison under different illumination conditions.

	TPR	SPC	PPV	NPV
Natural light	0.89	0.93	0.92	0.89
Extreme artificial light	0.82	0.86	0.85	0.84

Table 5.3: Test results on four post-development printed objects.

Colour	Metrics	TPR	SPC	PPV	NPV
Blue	Average	0.99	1.00	1.00	0.99
	Majority Vote	1.00	1.00	1.00	1.00
Wooden	Average	0.87	0.86	0.85	0.88
	Majority Vote	1.00	1.00	1.00	1.00
Dark Green	Average	0.81	0.85	0.84	0.82
	Majority Vote	0.98	1.00	1.00	0.98
Skin	Average	0.93	0.99	0.99	0.94
	Majority Vote	1.00	1.00	1.00	1.00

Table 5.4: Test results on objects printed with unseen material colours.

Colour	Metrics	TPR	SPC	PPV	NPV
Blue	Average	0.95	0.97	0.97	0.95
	Majority Vote	1.00	1.00	1.00	1.00
Wooden	Average	0.79	0.76	0.76	0.79
	Majority Vote	0.87	0.83	0.82	0.87
Dark Green	Average	0.74	0.82	0.80	0.76
	Majority Vote	0.91	0.94	0.94	0.91
Skin	Average	0.65	0.8	0.81	0.74
	Majority Vote	0.72	0.98	0.98	0.76

Table 5.5: An active learning approach on transparent purple objects.

Colour	Metrics	TPR	SPC	PPV	NPV
Image a	pre-active	0.48	0.68	0.58	0.59
	active	0.63	0.74	0.69	0.68
Image b	pre-active	0.69	0.75	0.72	0.72
	active	0.76	0.80	0.78	0.78
Image c	pre-active	0.58	0.64	0.60	0.62
	active	0.95	1.00	1.00	0.95

Experiment 4

3D printed watermarking retrieval is an application well suited for the adoption of an active learning approach where we expand the training set with examples on which the current network underperforms, and then we retrain. Indeed, on the one hand, performance assessment of the ability of the algorithm to retrieve the correct bit values can be done entirely automatically, on the other hand, the expansion of the training set requires the hand-annotation of images of watermarked objects, which is a laborious, tedious process. Therefore, being able to choose suitable images for inclusion to the training set is a matter of importance.

From Table 5.1, we notice that transparent purple is a particularly challenging material colour. We printed two more watermarked objects on this colour, capture 15 images from each object and added them to the training set. Table 5.5 shows the performance of the algorithm, before and after retraining with the expanded set, on three the three transparent purple test images where it initially performed the worst. We notice a considerable performance improvement.

Other experiments

The current retrieval algorithm works on planar only surface watermarks. An extension of the approach to other watermark carrier surfaces seems to be a problem to surface reconstruction and rather orthogonal to the machine learning part of the algorithm on which this paper concentrates. To test this claim, we printed similar

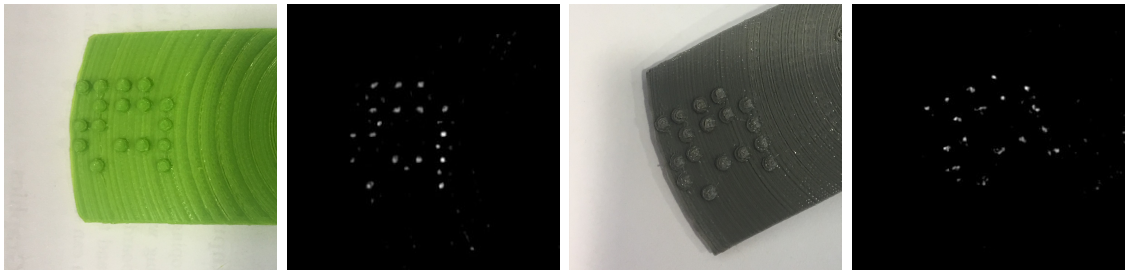


Figure 5.7: Confidence maps from non-planar watermarked surfaces.

watermarks on spherical surfaces and processed their images with CNN-3DW. In the examples shown in Figure 5.7, we notice that good quality confidence maps are indeed constructed.

In a final test, on each of the 400 bits positions, we computed the accuracy rate overall 240 images of Experiment 1. We found a weak but not negligible correlation between accuracy rate and distance from the centre of the information matrix with a Pearson coefficient of 0.46. The farther away from the information matrix centre a bit is, the higher the probability to be correctly retrieved. While we do not have an intuitive explanation for this result, we notice that the algorithm performs better over the less congested parts of the watermark, that is, near the boundary and the corners.

5.4 Conclusions and Future Work

We presented a method for retrieving digital information embedded as planar a surface texture on a 3D printed object. The proposed method was tested extensively, and it was shown to be a technically viable alternative to DotCode. Moreover, by integrating the information embedding process into the manufacturing process, we address several security vulnerabilities of the current widespread practice of displaying DotCode printed as separate items.

5.4.1 Limitations and Future Work

In tailoring our algorithm, we imposed various limitations to the solution, most of them aiming at isolating to study the machine learning part of the watermark

extraction pipeline. Thus, the registration of the confidence map relies on the use of a second printing material of distinct colour to mark the corners of a rectangular region of interest. Moreover, as a convenient robust solution to the watermark orientation problem, we use four distinct corner symbols consisting of one, two, three and four small dots, respectively. In another limitation, we restricted ourselves to planar surfaces. While CNN-3DW generates good quality confidence maps of non-planar watermarked surfaces, the extraction of the watermark from the confidence maps is essentially a surface reconstruction problem, quite different from the bump detection part.

In the future, we plan to extend the current work into watermarks embedded on non-planar surfaces. As we discuss in Section 5.3, that extension should not require any significant modifications of the CNN-3DW presented in this paper, but rather the development of new methods for processing the confidence maps. In another future research direction, we plan to develop higher capacity watermarks making use of more complex surface textures as carriers. In this case, we expect that the accurate retrieval of the watermark will require the development of more complex machine vision tools.

Chapter 6

Watermark retrieval from 3D printed objects via synthetic data training

In chapter 5, we used real training data. For the generation of a good training set, the acquisition process would require the 3D printing of several objects, which can be expensive, and the hand-annotation of numerous images from each of those objects, which is an error-prone and laborious process. In this chapter, we address this problem by adding synthetic data to an initial training set consisting of real images. The process of generating these synthetic training data is guided by the relatively simple but effective Domain Randomization (DR) technique. The results demonstrate that the inclusion of synthetic DR data to the training set increases the generalization power of the network, which, as a result, performs better on images from previous unseen 3D printed objects.

6.1 Introduction

The hand-annotation of this type of training data, as in [153], has two known sources of error. Firstly, there is a human error introduced when the marked coordinates of the watermark bumps are not exactly on the centroid of the bump. Secondly, even if we assume semi-spherical bumps, as it is the case in our experiments, there

is a systematic error introduced when a Gaussian kernel is applied on the marked coordinates to generate a smooth confidence map. Indeed, unless we manually adjust the size of the Gaussian kernel over each training image, which in practice it is infeasible, the training confidence maps do not adjust to the scale of the object's image, i.e., assuming objects of the same size, to the distance of the camera from the object. Moreover, in what is the most intrinsic source of error in that process, when object images are taken from low angles, the bumps images deviate considerably from the circular shape and will always be over or under-covered by the circular support of the Gaussian.

To address the problems of human and systematic errors in the hand-labelling process, which confuse the CNN models and reduce their accuracy, we propose to use the CAD models of the 3D printed models, which exist since they are needed for 3D printing, to produce synthetic training images with accurate annotations automatically. Apart from increasing the accuracy of the annotations, the proposed solution also reduces the cost of creating real training data, i.e. printing materials and labour, which means that we can have more variability in terms of object colours, illumination conditions and camera distance and angle.

Our approach follows recently proposed cost-efficient solutions for bridging the gap between real and synthetic data using graphic generators such as UnrealStereo [154], which can generate photo-realistic synthetic data and the corresponding ground truths. It has been successfully applied in training neural networks for optical flow [155, 156], semantic segmentation and stereo estimation. The UnrealStereo requires designers at the artist level, which means a person who can transform design projects and concepts into breathtaking imagery. Hence the domain randomization [157] has been recently proposed to address this limitation. Domain randomization is currently used for robots to locate objects with simple shapes, forcing the network to focus on the essential features of the objects in the images rather than photorealism.

Contribution. Addressing shortcomings in Zhang *et al.* [153], which relies on a high cost, small, manually labelled real image dataset, in this paper we employ

domain randomization for watermark retrieval from 3D printed objects, training the CNN with synthetic image data and confidence maps. The use of synthetic data not only reduces the cost of creating the training dataset, but also eliminates the human and the systematic errors of hand-labelling. The confidence maps are now precise in that they correspond exactly to the watermark bumps areas, see Figure 6.1. The main contributions are summarized as follows:

- Application of the domain randomization method to generate synthetic data for a non-trivial problem, watermark retrieval from 3D printed objects. The practicability of our approach rests that no artistic input is required at any stage of the process.

- Experimental investigation of the effect of the parameters of the synthetic data.

Our experiments show that for images from previously unseen 3D printed objects, the highest precision and recall rates are achieved when we train the neural network with a combined dataset consisting of synthetic DR and real data. A series of further experiments show that the inclusion of the synthetic data in the training set increases the generalization power of the network, enabling it to learn the basic invariant features of the target objects.

The rest of the paper is organized as follows. In Section 6.2 we discuss related work; in Section 6.3 we introduce our synthetic data image generator and our Convolutional Neural Network-3D Watermarking (CNN-3DW); in Section 6.4 we test with real image data and explore how synthetic data support training; finally, we briefly conclude in Section 6.5.

6.2 Related Work

Synthetic dataset creation and domain randomization

The synthetic data methods support accurate ground truth annotations and are cheap alternatives to annotating images. They have been widely used in deep learning. Jaderberg *et al.* [158] and Gupta *et al.* [159] paste real text images to blank

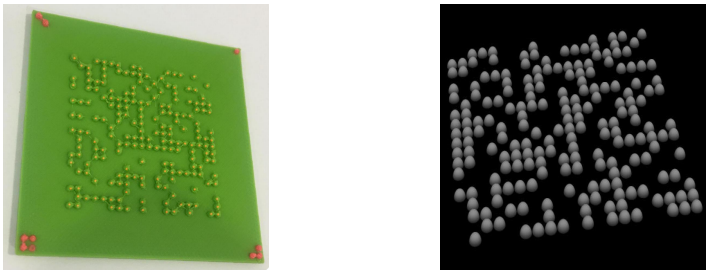


Figure 6.1: Real image with hand-annotated coordinates of the watermark centroids (left1). Ground truth image generated by the simulator (left2).

backgrounds and real natural background, respectively. Dosovitskiy *et al.* [160] combine renderings of 3D chairs images with natural image backgrounds, such as city, landscape and mountains, to train FlowNet. Handa*et al.* [161] build indoor scenes using CAD models to understand real-world indoor scenes. Atapour-Abarghouei and Breckon [162] train a depth estimation model using synthetic game scene data. Zhang *et al.* [154] develop a synthetic image generation tool to analyse stereo vision. The first three of the above methods use existing datasets and models, while the last two require artist level synthetic image data.

Domain randomisation is a simpler technique for training models on simulated images. It randomises some of the parameters of the simulator and generates a dataset of sufficient variety consisting of non-artistic synthetic data [157,163]. Here, we use domain randomisation on our own 3D watermarked models, as there are no such datasets available, and train a CNN to retrieve watermarks from images of the corresponding 3D printed objects.

6.3 Method

In this section, we describe the generation of the synthetic training dataset and present the proposed framework for automatic 3D watermark retrieval.

6.3.1 Synthetic Image Generator

To prevent the model from overfitting to unwanted features such as object colour, surface texture, bump size and shape, and camera viewpoint, the following aspects

were considered when creating the training dataset:

The 3D shapes. For simulating the expected diversity of 3D printed objects in a practical scenario, the generated 3D models should have a wide range of colours and textures, and be embossed with bumps of various sizes and shapes. Eventually, any real data would appear as a variation of some of the synthetic data, and their bumps will be identified correctly regardless of their peculiarities.

Camera viewpoint. Introducing training images captured from arbitrary view angles increases the robustness of the model, we rotate the virtual camera of the simulator around the model.

Scene illumination. In a practical scenario, when we capture images of 3D printed objects, we will not be able to control the illumination conditions. Thus we use multiple light sources, eliminating the influence of light intensity and shadow.

Scene background. By padding the captured images onto random backgrounds, we reduce the sensitivity to background noise. The background photos are sampled from available public datasets of indoor and outdoor scenes.

Software. For the synthetic data generation task, we adopted Pov-Ray [164], since it is open-source and supports several geometric primitives for constructive solid geometry, which is a popular modelling method for designing objects for 3D printing.

The setup of the synthetic image generator is demonstrated in Figure 6.2. It consists of five light sources, one 3D model, and one camera. When generating data, the camera revolves around the synthetic model to capture images under various angles; simultaneously, the synthetic model orbits light_1, i.e. the primary illuminating source placed in the middle, while rotating around its axis, to simulate variant beam directions. A random position shift of the camera within a moderate range is introduced further to increase the randomness of view points and scale, while additional illuminating sources lights_2-5 are introduced for simulating a complex illumination

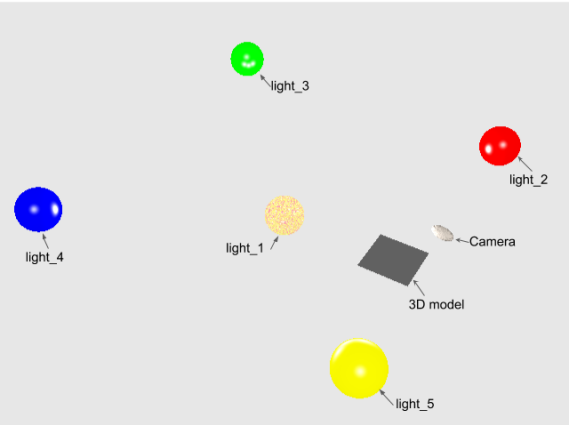


Figure 6.2: Synthetic image generator.

scenario. As to the 3D model itself, we introduce colour randomness and a wide range of optional textures to simulate the appearance of 3D printed objects. Note that the camera's orbital speed is slightly faster than the angular velocity of the rotating model.

We set Pov-Ray's float identifier clock at 120 frames per cycle with the initial frame at clock value 0.0 and the last frame at value 1.0.

Textures and colours. Model textures are randomly picked from Pov-Ray's files at various scales. The RGB colours are randomised at $\alpha * clock, \beta * clock, \gamma * clock$, where $\alpha, \beta, \gamma \in (0, 5)$ are random numbers.

Scene lighting. To approximate a realistic environment, we use five different light sources. The centre (light_1) is a parallel light source, and around it in four different orientations, we placed: a near point light (light_2), a conical spotlight (light_3), an area light (light_4) and a fading light (light_5).

Pose and Camera. Within each clock cycle, the synthetic 3D model rotates around its axis by $360 * clock * 30$ degrees and revolves around the central light by $360 * clock$ degrees. The camera revolves by $360 * clock * 12$ degrees, and its movement along the vertical axis is set to $jStart + jHeight * (1 - \cos(4\pi(clock - jStart)))/2$. The synthetic image is captured at pixel size $3072 * 4096$.

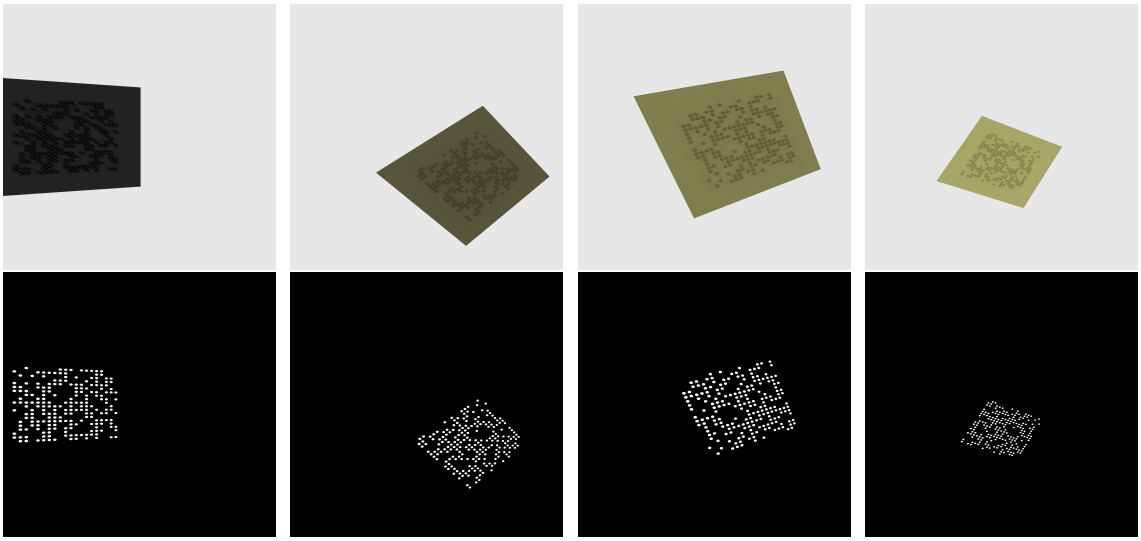


Figure 6.3: Synthetic image and corresponding ground truth at different clock values.

The corresponding annotations are generated simultaneously with the synthetic image, as shown in Figure 6.3. For generating annotations, we keep the same the camera pose and light parameters and change the watermark bumps to full white while the rest of the model is set to black. The produced synthetic annotation is an exact map of the bump regions.

In order to better enable network learning to be robustness. The random texture will be applied to the object of interest. Randomly insert different numbers of different types of lights, and render the scene from the viewpoint of a random camera. The resulting image with automatically generated ground truth labels is used to train the neural network. All these variants enable the network to achieve more complex behavior, but it is also very easy to implement.

Although our original (almost cartoon) images are aesthetically unsatisfactory, this obvious limitation can be regarded as an asset: our image creation speed is not only orders of magnitude faster (requires expertise, And less knowledge), and contains a variety of content. Forcing deep neural networks to focus on structural changes of important issues, rather than possible details, or may not exist in real images at the time of testing.

6.3.2 Watermark detector

Retrieving watermark information from 3D bumps is a dot detection problem. We tackle it by combining a deep convolutional neural network named **CNN-3DW** and an image registration module which processes the output of **CNN-3DW** to retrieve the watermark bit matrix.

CNN-3DW

The CNN-3DW is based on the network model shown in Figure 5.3. It consists of five convolutional layers with kernel sizes $9 \times 9, 7 \times 7, 7 \times 7, 7 \times 7, 1 \times 1$ and feature map channels 48, 96, 48, 24, 1, respectively. The Rectified Linear unit (ReLu) is used as an activation function. Each of the first two layers is followed by a standard max pooling layer, increasing the receptive field. For all the convolutional layers, we set the step size as one and do zero paddings so that the convolutional operations do not alter the image size. Since the model is fully convolutional, it can handle different input sizes. The reason we designed our network based on FCN rather than a deeper architecture such as VGG and ResNet is that we found in our experiments that the output of FCN was more stable. Indeed, locating dots on a single coloured plate is a relatively simple task and the adoption of deeper models is more resource expensive. Further, as shown in Figure 6.3, the ground truth image composes of mostly zero value pixels indicating the irrelevant background, and by stacking non-linearity active functions such as ReLU can easily cause the network model to predict all-zero.

Because our training data comes from inexpensive synthetic images with controllable parameters, data augmentation such as mirroring, translational shift, rotation and relighting is not necessary during the training phase. Instead, for time and resource efficiency, since the original images are oversized for graphics cards with small capacity, we randomly cropped each image to smaller patches of uniform size (1024*1024).

We implemented CNN-3DW in Pytorch and adopted the mean squared error (MSE) for computing loss. The model is optimised using Adam [150]. The initial learning rate is $1e-4$ and is decreased to $1e-5$ after 150 epochs. We used a single

NVIDIA GeForce GTX TITAN X graphics card to train our network with a batch size of 12.

Image registration, processing and matrix retrieval

Using the trained CNN-3DW model, we are able to estimate a confidence map $\mathbf{Y}$ of the watermark bumps in the image $\mathbf{X}$. To extract the embedded information matrix from the estimated confidence map $\mathbf{Y}$ is still non-trivial. Indeed, due to variability in the illumination conditions and surface textures inadvertently introduced by the popular, inexpensive printers based on Fused Deposition Modelling (FDM) technology, a large amount of background noise can be introduced, posing significant challenges to the automatic watermark region localisation. Since a significant bias in the watermark region localisation would lead to catastrophic error in the following stages, we decided to tackle this issue by printing four *landmarks* at the corners of the watermark with distinct colours (see Figure 6.4). During retrieval, the watermark regions are easily located by finding the differently coloured *landmarks* at the four corners.

Further, since we do not require from the user to align their camera with the 3D information matrix when capturing images, the generated confidence map needs to be normalized before retrieving the information. We use image registration [165] to transform the quadrilateral watermark region in the confidence map $\mathbf{Y}$ to a square region (see Figure 6.4), using Matlab's built-in tools to map the *landmarks* to the four corners of a square region. We denote the normalised watermark region of the confidence map as $\mathbf{Y}^r$ and process it to extract the embedded information matrix $\mathbf{M}$.

The registered confidence map $\mathbf{Y}^r$ is first binarized using a threshold value t.

As a final step, we extract the information matrix M from the coordinates of the regionprops centroids that have been assigned bit values 1. The point falling into the i-th and j-th clusters, respectively, will correspond to the (i, j) entry of the matrix $\mathbf{M}$.

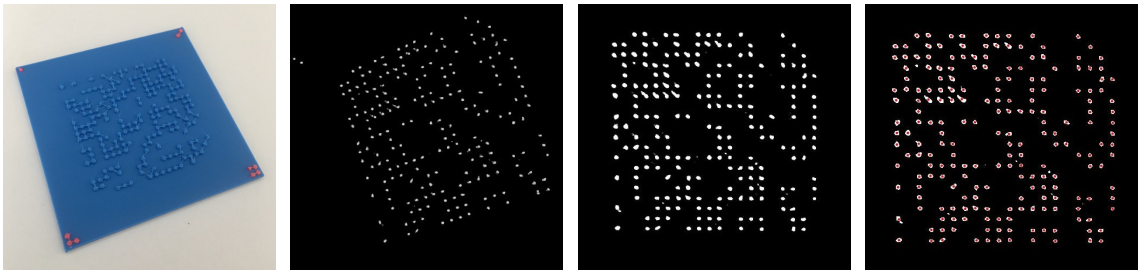


Figure 6.4: The test image (left1); the confidence map outputted by CNN-3DW (left2); the binary confidence map obtained after regularisation followed by thresholding (left3); the detected centroids of bit value 1 regions (left4)

6.4 Experiment

To assess the effectiveness of training the neural network model with synthetic data, we conducted a series of experiments and analyzed the results, evaluating the performance of the proposed technique in various situations.

6.4.1 Training datasets: 3DW-real and 3DW-syn

For our 3DW-syn dataset, we generated 2K images each one carrying a randomly generated 20×20 watermark. As described in section 6.3, the model of the watermarked object is assigned a random colour and texture. Scenes are rendered under random camera poses and illuminated from light sources of various types. For comparison, we used a hand-labelled dataset (3DW-real) of 255 images of watermarked 3D printed objects [153]. Although we use more synthetic than real images, the cost of creating the 3DW-syn dataset was lower, and 3DW-real was augmented in more ways than 3DW-syn. In particular, we applied random crop, brightness and contrast setting to both datasets, while 3DW-real was also augmented with random flips and random resizing. In all cases, training was stopped when the validation error was stable, and the best weights were saved.

For testing, we used 35 images (Test_A) which are taken from same 3D printed models used in 3DW-real, but with different illumination conditions and camera poses, and 20 images (Test_B) taken from other 3D printed models made from materials that were not used in the 3DW-real dataset. Detection performance was eval-

uated using recall TPR=TP/(TP+FN) and precision PPV=TP/(TP+FP), wherein our case TP and FN are the numbers of value 1 bits retrieved correctly and correspondingly incorrectly, while FP is the number of value 0 bits retrieved incorrectly.

6.4.2 Experimental Setup

We evaluate the watermark retrieval capabilities of the proposed technique under the use of various training datasets and provide an ablation study to explore further the effectiveness of using synthetic data.

Watermark retrieval

We evaluate the performance of the proposed framework on three scenarios: neural network model trained with real data (3DW-real) only, synthetic data (3DW-syn) only, and the combined dataset (3DW-real + 3DW-syn). Precision and recall are tested separately on Test_A and Test_B, and the results are shown in Table 6.1. Note that Test_A consists of images from the same 3D objects as the images of the (3DW-real) training dataset, while Test_B contains images from different objects.

Figure 6.5 shows indicative outputs of the CNN-3DW on Test_A and Test_B dataset, after being trained on either of the three training datasets we used. Notice that even though our network has never seen a real image when trained on (3DW-syn) only, it is able to detect most watermark bumps successfully. Moreover, when the network was trained on combined (3DW-real + 3DW-syn), on the most challenging Test_B, it outperformed the network trained on (3DW-real) by a 15% in recall value and 5% in precision. This surprising result illustrates the power of such a simple technique for bridging the reality gap and rectifying the human and systematic error disadvantage in manual labelling.

Next, we study the effect of fine-tuning [167] the number of real images we add to (3DW-syn) to create a combined training dataset. Results with varying percentages of (3DW-real) added to the training dataset are shown in Figure 6.6. We note that when a combined dataset is used, recall and precision increase with the number of real-images included. However, when less than 70% of (3DW-real) is included, the recall value is lower compared to training with (3DW-syn) only. We hypothesize

Table 6.1: Performance comparison among training datasets: real-world dataset, synthetic dataset and their combination, on Test_A and Test_B.

Dataset	Test_A		Test_B	
	Recall	Precision	Recall	Precision
Real-world dataset	**0.85**	**0.89**	0.57	0.75
Synthetic dataset	0.65	0.53	0.62	0.55
Combination	0.74	0.83	**0.72**	**0.80**

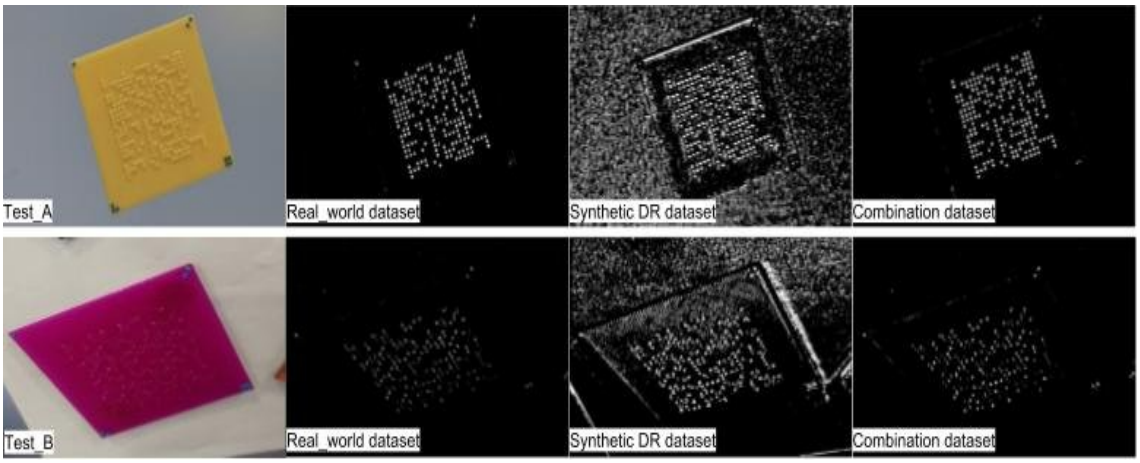


Figure 6.5: Confidence maps of images in Test_A and Test_B using CNN-3DW trained on: real-world dataset (left2), synthetic dataset (left3) and their combination (left4).

that when (3DW-syn) is mixed with few real images only, the real images confuse the neural network, preventing the learning of features from (3DW-syn).

Ablation study

To study the effect of individual DR parameters, we conducted an ablation study. In all cases, we train with synthetic data only, and since no real images are used, we can safely use the larger Test_A for testing. Figure 6.7 shows the results of controlling individual components of the DR data generation procedure as described in more detail below.

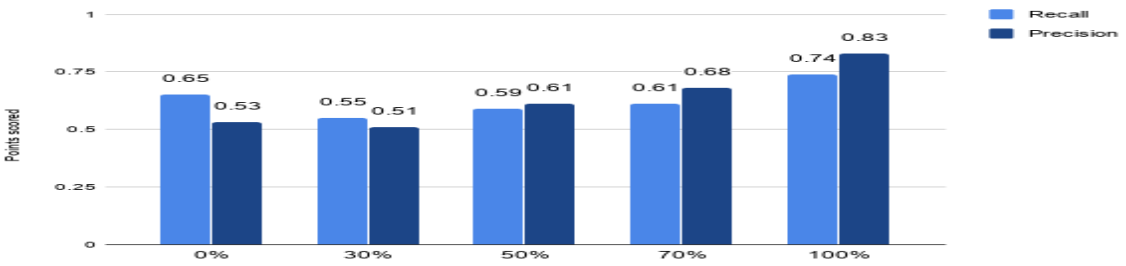


Figure 6.6: Percent of real images used in fine-tuning.

Training dataset size. In this experiment, we train with a percentage only of (3DW-syn), keeping all other parameters unchanged, and we study the effect of the training dataset size upon performance. The results show that recall and precision increase with the size of the dataset, but after a certain level, recall and precision rise little.

Texture. In this experiment, keeping other parameters unchanged, we create two sets of 1.5K synthetic images each, one utilizing half only of the available texture types, the other utilizing the full set of available textures. When the textures types applied on the watermark bumps were limited, recall and precision dropped to 0.49 and 0.45.

Data augmentation. In this experiment, during training with the full (3DW-syn), we turned off image brightness and contrast data augmentation. The recall falls to 0.58 while the precision barely drops by 0.02. Note that, in this case, the effect of lighting adjustments on already generated images is smaller than the effect of a changing a simulator parameter such as the number of texture types available.

Training strategies

Finally, we studied the effect of pre-trained weights upon the performance of CNN-3DW. In this experiment, we initialized the weights of the CNN-3DW network using random weights, the weights of the (3DW-real) trained network, and the weights of the (3DW-syn) trained network, respectively. In all cases, we retrained on (3DW-real + 3DW-syn) and tested on Test_B. The results are shown in Table 6.2.

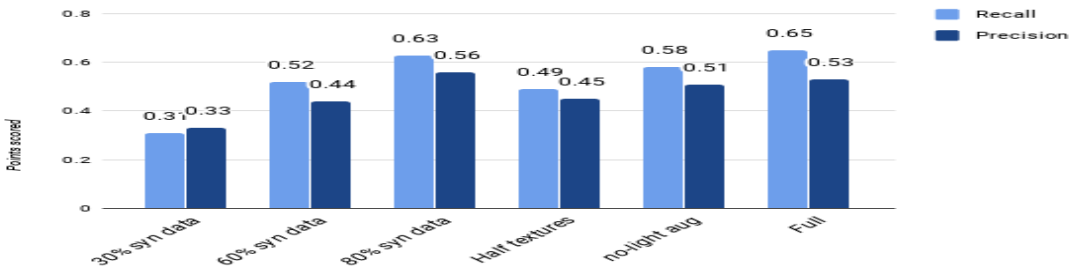


Figure 6.7: The effect of controlling individual components of the DR data generation procedure.

Table 6.2: Training on (3DW-real + 3DW-syn) with three sets of pretrained weights: random and obtained by training on (3DW-real) and (3DW-syn), respectively. Testing was done on Test_B.

	Random weights	(3DW-real) weights	(3DW-syn) weights
Recall	**0.72**	0.58	0.61
Precision	**0.80**	0.69	0.75

First, note that the (3DW-real) weights achieve recall and precision of only 0.58 and 0.69 respectively, showing that the real-image weights are not very useful for training a network to operate on the Test_B dataset. It is, in fact, a significant challenge with networks today, namely, that they often fail to transfer from one dataset to another. In contrast, the (3DW-syn) weights achieve better recall and precision rates, indicating that synthetic data enable the network to learn features which are invariant to the specific dataset. Finally, the best rates are achieved with random weights, i.e. when synthetic DR data are mixed with real-image data and train the network without pre-training. Indeed, training with synthetic data only produces inferior results, while training with real data only does cope well with variability in parameters such as printing materials.

6.5 Conclusion

We showed that domain randomization is a useful technique in the context of watermark retrieval from 3D printed objects. Using synthetic DR data, intentionally

disclaiming photorealism to force the neural network focus on the most relevant features, we trained the neural network (CNN-3DW) to generate confidence maps for the locations of the watermark bumps. When synthetic DR data were mixed with real image data, the retrieval rates were higher than using real or synthetic data alone.

While the process of inferring information about 3D objects from 2D images of them would always be subject to certain inherent limitations, in some applications recourse to the exact CAD models of those 3D objects can facilitate computer vision tasks. In particular, the annotation of the synthetic images of our training dataset is automatic, eliminating the human error and cost of a hand-labelling, and performed in the 3D domain, eliminating a systematic error that is unavoidable in certain situations where information about 3D objects is directly annotated on 2D images. In the future, we plan to test the technique on more general problems of object recognition, object location and counting problems, working with more complex instances of 3D printed objects and their corresponding CAD models.

Chapter 7

A study of the effect of the illumination model on the generation of synthetic training datasets

Chapter 6 demonstrated that the use of computer-generated images to train DNNs is an attractive alternative to real images, especially since the latter are scarce and expensive. In this chapter, we move away from 3D printing as the object of our study. Instead, we use 3D printing as a means to study how the illumination model of the rendering software affects the quality of the computer-generated images and their suitability for training DNNs. We compare several illumination models on three different network architectures, ResNet, U-Net, and a novel combined architecture. The results show that the effect of the illumination model is important, comparable insignificance in terms of DNN performance to that of the network architecture.

7.1 Introduction

To the best of our knowledge, this relationship has not been studied in the literature. In particular, there are no systematic comparisons of the performance of DNNs when different illumination models are used for generating a synthetic training dataset.

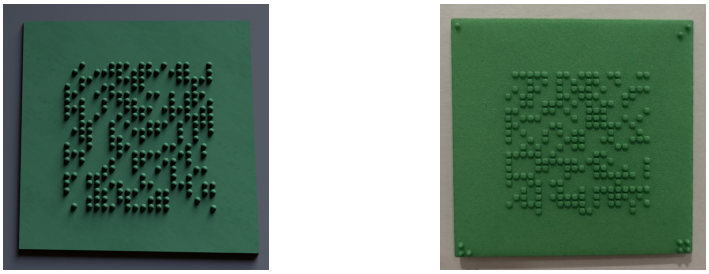


Figure 7.1: A synthetic training image (left1) and a test image (left2), which is the photo of a 3D printed object.

Indeed, in most cases, the illumination model becomes a convenience choice dictated by the capabilities of the rendering software. Quite often, the model is not even reproducible, as it is not adequately described in the paper.

Here, in a systematic study of the influence of the illumination model, we compare the DNN performance over eight training datasets, generated with different illumination models. Five of them use different light probe images, i.e., models of outdoors or indoors natural lighting environments captured by specialised sensor systems at various resolutions. The light probe image is HDRI (high dynamic range imaging) environment map, which is a type of light source that is well-suited for representing "natural" illumination. Additionally, We also use three synthetic illumination models of increasing levels of complexity.

The tested for the comparison of the various illumination models is watermark retrieval from 3D printed objects [153]. The watermarks are embedded on flat surfaces of 3D printed objects and encode information in 20×20 bit arrays, where a bit value equal to 1 corresponds to a semi-spherical bump on the object's surface, Figure 7.1. In a first stage, the retrieval algorithm, which decodes the watermark from one or more photos of the physical 3D printed object, uses a DNN to generate a confidence map of the location of the bumps and then uses a standard image processing pipeline to extract the bit-array from the confidence map. The overall performance of the retrieval algorithm mainly depends on the segmentation performance of the DNN for watermark bumps.

The use of 3D printed models for all testing aims at neutralising the influence of the geometric modelling part of the graphics pipeline. Indeed, the same digital 3D

models that are used for generating the synthetic training images are also used for 3D printing the physical objects, making the underlying 3D geometries in training, and the test sets almost identical. Since differences in the geometry between training and test sets are expected to contribute significantly to the DNN error, their elimination reduces the risks to the validity of the results. To reduce the bias introduced by the architecture of the DNN, we experimented with three different networks; ResNet50 [66], which gives low noise confidence maps, a variant of U-Net [72], which gives proper localisation, and a combination of the two networks we developed, which to some extent combines the strengths of the previous two networks.

Contribution: We present a systematic study of the relationship between the illumination model and the quality of synthetic datasets generated for DNN training. The main findings can be summarised as follows:

- The resolution of the light probe is a critical factor regarding its suitability for synthetic data generation. The similarity between the light probe and the illumination of the test environment is also important.
- Synthetic illumination models can give results that are competitive to those obtained from light probes. However, the right level of fidelity towards the test environment is required, as well as.

As a secondary contribution, we note that a characteristic of our approach, linked to the use of 3D printing technology, is that the geometries of the digital objects in the training set and the physical objects in the test set are identical; they correspond to the same CAD models. In this new setting, we can confirm and illustrate interesting characteristics of DNN behaviour, in particular, that ResNet50 generates confidence maps with lower noise, while U-Net outperforms ResNet50 in terms of localisation.

7.2 Background

7.2.1 Deep neural network

DNNs are the current state-of-the-art in various fundamental computer vision problems such as object detection [62,168–170,174] and semantic segmentation [171,172].

The various extensions of Convolutional Neural Networks (CNNs), such as Fast R-CNNs, R-FCNs and SSDs are widely considered as some of the best performing networks for object detection. In [173], the use of a classical model, Fast R-CNN, was proposed for deep learning-based object detection. The Fast R-CNN is an end-to-end training object detection framework that uses multi-task loss on each labelled Region of Interest (RoI) to train for classification and bounding-box regression jointly. Moreover, unlike SSPnet [176], Fast R-CNN training can update all network layers. In [174], a region-based fully convolutional network (R-FCN) was proposed for objects detection. The R-FCN considers each region proposal, divides it up into sub-regions and iterates over those sub-regions. In this process, the R-FCN is able to use each generated position-sensitive score map to encode the position information with a relative spatial position. R-FCNs have higher accuracy and are several times faster than Fast R-CNNs. In [168], a single-shot detector (SSD) was proposed, processing the images to simultaneously estimate bounding boxes for the detected objects and predictions for their class, that is, the SSD does not only use the network to generate RoIs, but simultaneously classifies those regions.

Other popular neural network models tend to be fairly similar to the three described above, and they mostly rely on very deep CNN's architectures, such as ResNet [66]. In [66], the deep residual learning technique was introduced, and ResNet was proposed as a first example implementing that architecture. Its main novelty was the introduction of residual links, shortcut connections enabling cross-layer connectivity, facilitating the convergence of the training process by avoiding gradient diminishing problems. To this day, ResNet remains one of the most powerful network architectures available.

U-Net was proposed in [177] as an alternative to very deep CNN architectures such as ResNet. It is a lighter network sharing many common features with FCNs. The main difference between FCNs and U-Net is the latter's symmetric architecture, which concatenates the feature maps in the expanding path with the corresponding cropped feature maps in the contracting path, while, in contrast, in FCNs feature maps are summed.

The ResNet has a large receptive field, which means that the network retrieves

more coarse information and can ignore tiny details [66]. In contrast, due to the skip connections, U-Net is good at retrieving information corresponding to the image details [177], making it a popular choice in the medical domain. As illustrated in [178] on a typical skin analysis application, the outputs of the U-Net show more details and give higher IOU rates than those of ResNet50, but also contain more noise.

7.2.2 Synthetic dataset generation

Synthetic dataset generation is an efficient alternative to the use of natural images in DNN training, especially when natural images are scarce, or expensive to obtain, or their annotation requires extensive expert input and is thus costly. On the other hand, synthetic data generation might have a quite steep overhead in the construction of the 3D models, but then the generation of abundant synthetic data becomes a quite straightforward and low-cost process. Moreover, in various application scenarios, annotations serving as ground truths not only can be automatically created at a minimal additional cost, but most importantly, they can be exact, free of any human-introduced errors.

Synthetic data have been widely used in training deep learning networks. Synthetic datasets are generated by pasting text images are on black and various natural backgrounds, respectively [158, 159]. Renderings of 3D chair images are combined with natural image backgrounds to train FlowNet [160]. Indoors scenes are built using CAD models to train a network for indoors scene understanding [161]. A depth estimation model is trained using synthetic scene data generated by a game engine [162]. Synthetic images train a network for stereo vision understanding [154]. The first three of the above methods used existing image datasets and 3D models, while the last two utilised purpose-built artist level modelling.

Domain randomisation is a simpler technique for training networks on simulated images. It randomizes some of the parameters of the simulator and generates a dataset of sufficient variety consisting of non-artistic synthetic data [157,175]. Here, we borrow from the domain randomisation technique in the way we treat some other important rendering variables that are not directly related to the illumination model,

specifically, texture and camera position and orientation.

7.3 Experimental setup

In this section, we first describe the test dataset, consisting of images of 3D printed objects captured by a mobile camera. Next, we describe the generation of the synthetic training datasets, focusing on the illumination model, which is the experimental variable we study here. Finally, we describe the architecture of the neural networks of the experiment.

7.3.1 Test dataset

We 3D printed eight watermarked objects in total, seven of them on the plastic of various dark and bright colours and one in metal. Each object was printed on a single material and colour. The watermarks are 20×20 arrays of randomly generated bits, encoded by semi-spherical bumps, the size of which varies from object to object. Each printed object carries a different watermark to ensure that high watermark retrieval accuracy means that the network can successfully detect and localise bumps on the object's surface, rather than memorise their relative locations. Figure 7.2 shows the eight printed watermarked objects.

The test dataset consisted of 20 images from each object, 160 in total, which means that each neural network/training dataset combination was tested on the detection and localisation of $160 \times 400 = 64,000$ possible bumps. The images were taken indoors, under natural light coming from windows and artificial light from the ceiling. The background was monochrome and had a fine texture, ensuring that any background textural features were smaller than the watermark bumps. The images were shot from various camera positions: the distance was roughly $1m$, the horizontal angle varied from $0°$ to $360°$, and the vertical angles, expected to have the most significant effect on the performance of the retrieval algorithms, varied from $90°$ (looking at the object from the top) to about $45\circ$. The camera was pointing to the target 3D printed object, with small errors introduced in purpose. Figure 7.3 shows two test images, one from a large and one from a small vertical angle.

Figure 7.2: The 3D printed objects.

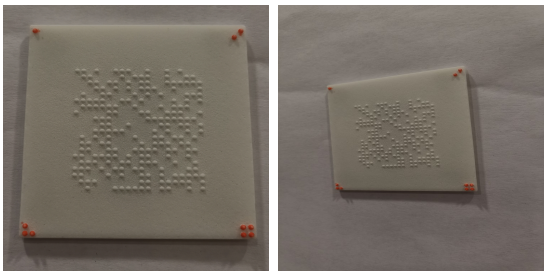


Figure 7.3: Test images from a high and a low vertical angle.

Table 7.1: Illumination models.

	Illumination model	Resolution
1	Indoor dining room	3072×6144
2	Indoor cathedral	1536×3072
3	Outdoor glacier	1024×2048
4	Indoor kitchen	640×640
5	Outdoor, covered hallway	640×640
6	Sky, physically based skylight at 10am	synthetic
7	Indoor office, area lights	synthetic
8	Indoor classroom, area lights and skylight	synthetic

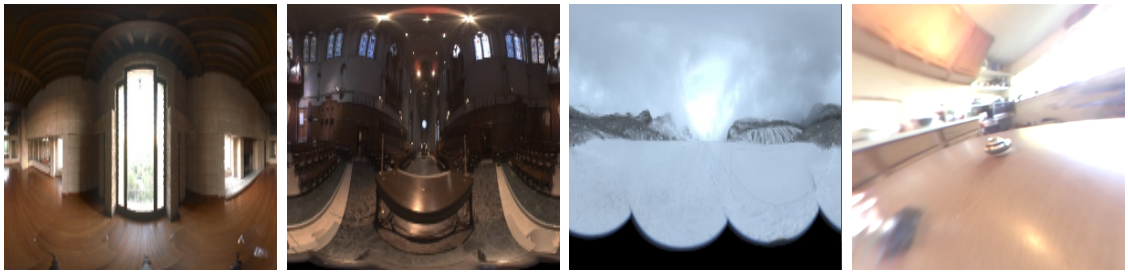
7.3.2 Synthetic Image generator

The training dataset consists of rendered images of the CAD models of the 3D printed objects. We note that the levels of noise introduced by the current, consumer-level 3D printing processes, can be significant, and part of the error in the watermark retrieval can be attributed to imperfections of the 3D printing process. Nevertheless, by choosing a commercial 3D printing service, rather than in-house printing, and by sending to the printers the exact 3D models that were used in the generation of the training sets, we neutralised the effect of 3D geometry as much as possible.

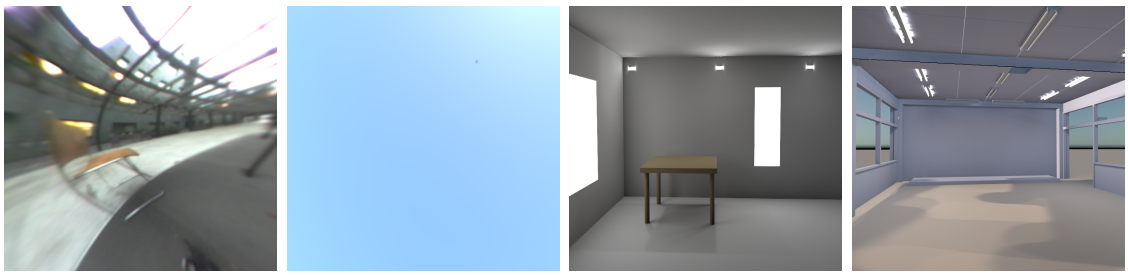
For all rendering, we used Mitsuba [179], which, being an open-source software with over 100 plug-ins ranging from textures and materials to light sources, supported all the variables of our experimental setup as described below.

Illumination model: The illumination model used in the generation of the training dataset is the main variable of study in the experiment. Table 7.1 lists the eight illumination models that were used to create the corresponding eight training datasets.

The first five models are based on light probes, i.e. images recording the incident light at a particular point in space. Figure 7.4 (a-e) shows the five light probes we used, downloaded from [180] and [181]. They are at various resolutions, some capture



(a) Indoor room (b) Indoor cathedral (c) Outdoor glacier (d) Indoor kitchen



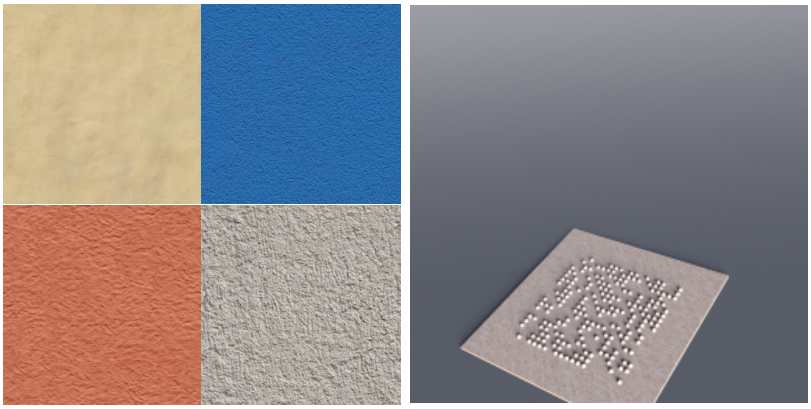
(e) Outdoor hallway (f) Sky, at 10am (g) Indoor office (h) Indoor classroom

Figure 7.4: The illumination models of the training sets.

outdoor and other indoor environments, and have various levels of similarity with the illumination conditions under which the test set was acquired. In particular, that similarity is high in the indoor dining room (#1), indoor kitchen (#4), and to some extent the covered hallway (#5), while the cathedral (#2) and the glacier (#3) have their one distinct illumination models.

The three synthetic illumination models are shown in Figure 7.4 (f-h). The first is simulated 10 am skylight generated by the physically based method proposed in [182], and the scene is void of any geometry. The second, model is the model of an office room with large area lights simulating windows and some secondary wall lights, which nevertheless do not correspond to a feature of the test dataset environment. Finally, the third illumination model consists of a faithful 3D model of the test room, faithful ceiling lights, and 10 am skylight entering from transparent windows. As the light probes and scene lights are imported directly, we directly use default parameters.

Textures: For simulating the colour and material diversity of the test objects, the generation of the training dataset is based on a wide range of colours and textures.



(a) Texture map samples (b) Synthetic image sample

Figure 7.5: Texture map samples and training dataset image sample.

The diffuse map is built upon the bitmap with additional diffuse reflectance, using the trilinear filter. To account for the variety of colors and materials in the 3D printed objects, we built a texture library containing 100 different texture images found using an online search engine. In addition, a monochrome planar background, which is rendered in gradient colour, is placed behind the watermarked 3D model to reduce the effect of the reflections.

Camera position: For each image, the camera position is defined by a triplet of random numbers (d, θ, ϕ). The camera is placed at a distance $d \in [50cm, 100cm]$ above the plane of the object, looks at the object from a vertical angle $\theta \in [45°, 90°]$, and revolves around the object by $\phi \in [0°, 360°]$.

7.3.3 Watermark detector

The first component of the watermark retrieval algorithm is a neural network returning a confidence map encoding the probability a pixel is part of a bump, rather than the background. We used an adaptation of the U-Net network, which we called **Unet-3DW**, ResNet50, and a combination of the two. The following components of the algorithm are an image registration module followed by a module that processes the registered confidence maps to retrieve the watermark bit matrix. Figure 7.6 shows the flow diagram for the case of the combined neural network.

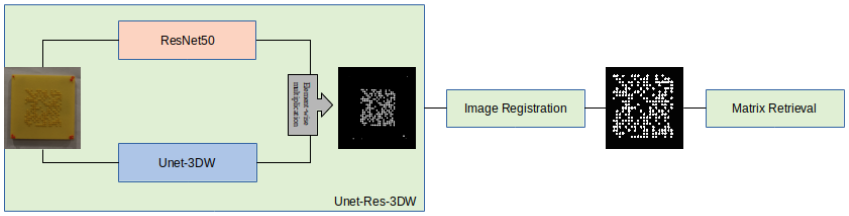


Figure 7.6: The flow diagram of the combined neural network.

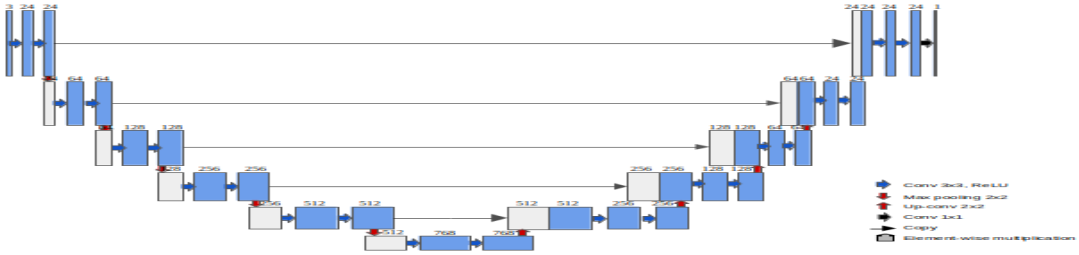


Figure 7.7: The Unet-3DW architecture.

DNN architectures

The standard U-Net model has four downsampling and four upsampling steps, each one halving or doubling the number of feature channels, respectively. In contrast, our **Unet-3DW** architecture consists of six downsampling and six upsampling steps, again laving and doubling the number of feature channels. In the contraction, each layer contains two repeated 3×3 padded convolutions and a 2×2 max pooling with stride 2, where each convolution is followed by a batch normalisation (BatchNorm2d) and a rectified linear unit (ReLU). The central bottleneck only contains one convolutional layer. In the expanding part, each layer consists of two padded 3×3 convolutions, each one followed by a ReLU. The first convolution concatenates two parts: the first comes from the corresponding feature map of the contraction path, and the second is a 2×2 up-convolution of the feature map. The second is a 3×3 convolution. At the very final layer, each 24 feature vector is mapped to the desired confidence map by a 1×1 convolution.

The second network we employed is the **ResNet50**. The third DNN we employed, combines and balances ResNet and U-Net, see Figure 7.6. As stated, for example, in [183], the deep layers of a CNN learn high-level global features, which

contain the most information about global patterns in the images, while low layers extract more local features. The combined network learns both global and local features, generating a confidence map by element-wise multiplication of the ResNet50 and Unet-3DW outputs. It is fed to the last layer with a LogSigmoid activation, which generates the final confidence map of the combined DNN.

DNN implementations

We implemented both ResNet and UNet-3DW in Pytorch and adopted the $BCEWithLogitsLoss$ for computing their loss. The loss combines in a single class a sigmoid layer and the $BCELoss$, and is numerically more stable than using $BCELoss$ only.

$$\begin{aligned} &BECWithLogitsLoss(x, y) = \\ &\frac{1}{N}\sum_{i=1}^{N}[y_i \times \log(\sigma(x_i)) + (1 - y_i) \times \log(1 - \sigma(1 - x_i))] \end{aligned} \tag{7.3.1}$$

where x is the predicted value, y is the target value and N is the total number of pixels.

The combined model is trained end-to-end with a joint loss function L:

$$L \quad = \quad \alpha \times L_{Unet-3DW} + \beta \times L_{ResNet} \tag{7.3.2}$$

where α and β are used to balance the weight of the two branches, and we found that $\alpha = 1$ and $\beta = 0.8$ works well for the problem at hand.

The networks are optimised using Adam [150]. The initial learning rate is $1e-3$ and is decreased by $1e-1$ after every 100 epochs until $1e-5$. We used a single NVIDIA GeForce GTX TITAN X graphics card and trained our network with a batch size of 4.

Because our training data are inexpensive synthetic images with controllable parameters, data augmentation such as mirroring, translational shift, rotation and relighting is not necessary during the training phase. Instead, for time and resource efficiency, we render our synthetic image with 1024×1024 size, which is lower than the 3072×2048 of the test images.

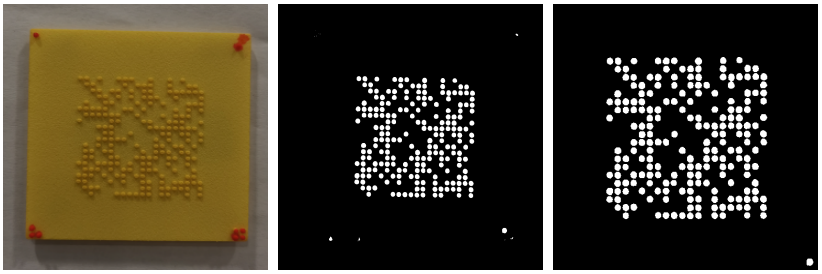


Figure 7.8: Test image (left1), confidence map outputted by CNN-3DW (left2), binary confidence map obtained by regularisation followed by thresholding (left3).

Image registration and watermark matrix retrieval

The extraction of the embedded watermark matrix from the estimated confidence map is still a non-trivial task.

Since a significant error in the localisation of the watermarked region would lead to catastrophic error in the following stages, we decided to tackle this problem by annotating four landmarks at the corners of the watermark region with a distinguishable colour (see Figure 7.8). During retrieval, the watermark regions are easily located by finding the differently coloured landmarks at the corners, and use them to transform the quadrilateral watermark region into a square region (see Figure 7.8). The registered confidence map is then binarised by Otsu thresholding. We use Matlab's *regionprops* function to detect its connected regions and obtain estimates of their centroids and their two semi-axes. If the sum of the two semi-axes is above a threshold, a bit value 1 is assigned to the centroid of that region, see Figure 7.8.

As a final step, we need to extract the watermark matrix M from the coordinates of the region-props centroids that have been assigned bit values 1. Ideally, there would exist only m (the number of rows and columns of $\mathbf{M}$) unique values for the x and y coordinates. However, due to imperfections in the previous steps, the coordinates have biases. Instead, we use K-means clustering on the x and the y coordinates of the centroids, respectively. We set the number of clusters as m and rank the m cluster centres. The point falling into the i-th and j-th clusters, respectively, will correspond to the (i, j) entry of the matrix $\mathbf{M}$.

7.4 Experiment

To assess the effect of the illumination model on the quality of the generated synthetic data, we conducted a series of experiments on several network/training dataset combinations.

7.4.1 Training datasets and evaluation metrics

We generated eight training datasets of 2K images each, one for each illumination model. Each image carries a distinct, randomly generated, 20×20 watermark, and a randomly chosen texture from the set of 100 different textures, as described in section 7.3. The camera positions are unique within each dataset, that is, images are captured under 2K different camera positions, but the same set of camera positions is used in all eight datasets.

The performance metrics we report are the recall TPR (True Positive Rate) =TP/(TP+FN), precision PPV (Positive Predictive Value) = TP/(TP+FP) and F1 scores 2×PPV×TPR/(PPV+TPR), where TP and FN are the numbers of bits with value 1 retrieved correctly and correspondingly incorrectly, while FP is the number of bits with value 0 retrieved incorrectly.

7.4.2 Results

In section 7.4.2, we evaluate the watermark retrieval capabilities of the three networks we tested, aiming at selecting an effective retrieval algorithm. In section 7.4.2, we selected algorithm is used to explore the effect of the illumination model on the quality of the generated synthetic datasets.

Assessing DNN performance

We present results for each of the eight test sets separately, each test set containing photos of a particular 3D printed object. On each test set, we tested $3 \times 8 = 24$ network-training set combinations, and for each of the three networks, we report the average F1 score over the eight training sets. Table 7.2 summarizes these average F1 scores for Unet-3DW, ResNet50, and the combined Unet-Res-3DW. Note that

all training data are synthetic images and all test data are real photos of 3D printed objects.

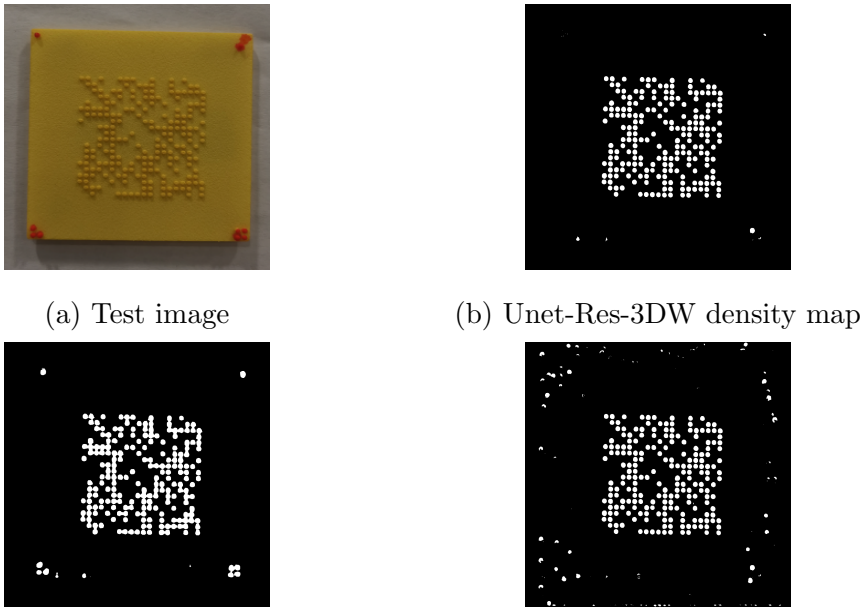
Table 7.2: Average F1-scores for Unet-3DW, ResNet50 and Unet-Res-3DW.

	Unet-3DW	ResNet50	Unet-Res-3DW
red	0.78	0.76	**0.81**
yellow	0.84	0.79	**0.89**
green	0.83	0.82	**0.90**
white	**0.88**	0.84	**0.88**
black	0.77	0.72	**0.83**
blue	0.77	0.79	**0.83**
purple	0.72	0.74	**0.75**
metal	0.76	0.73	**0.80**

From Table 7.2, we notice that the combined Unet-Res-3DW outperforms both of its components. We believe that the higher performance of the combined network can be partly attributed to the complementarity of the information retrieved by its two components. Figure 7.9 shows the density maps computed by the three networks on a test image from the yellow 3D printed object. We notice that as a result of ResNet50 focusing on deep and abstract features, its density map contains very little noise. However, the clusters of high-valued pixels corresponding to the object's bumps tend to overlap and get connected, rather than being clearly outlined and distinct. In contrast, in the density maps from the Unet-3DW, which excels on retrieving local information, there is more noise, but the high-valued pixel clusters corresponding to the object's bumps are well-localized and clearly outlined. The density map of Unet-Res-3DW seems to retain the strengths and avoid the weaknesses of its two components, as it avoids excessive noise and simultaneously produces well-localized clusters of high-valued pixels representing the object's bumps.

Finally, we note that the performance of the three networks does not seem to drop on the white and the black 3D printed objects, even though the texture library does not contain these two colours, and that they also cope reasonably well on the metal model, despite its very different reflectance properties.

Table 7.3 gives a more detailed description of the performance of the Unet-Res-



(a) Test image (b) Unet-Res-3DW density map

(c) ResNet50 density map (d) Unet-3DW density map

Figure 7.9: Density maps of Unet-Res-3DW, ResNet50 and Unet-3DW.

-

3DW network by reporting separately the average precision and recall separately over the eight training sets, as well as the performance of an ensemble algorithm determining the value of each bit of the 20×20 watermark matrix through a majority vote over 20 different images of the 3D printed object. We notice that majority vote leads to significant increases of both precision and recall, and also we note that the requirement for access to multiple images is naturally satisfied in our application scenario of watermark retrieval by a mobile phone camera.

Assessing the effect of the illumination model

In the study of the effect of the illumination model, each of the eight test sets was tested against 8 network/training set combinations, the network always being the Unet-Res-3DW in ensemble mode, and the training sets corresponding to the eight illumination models in Table 7.1, shown in Figure 7.4. The results are summarized in Table 7.4.

From Table 7.4 we see that, as expected, the choice of illumination model for generating synthetic data can have a significant effect on the performance of the

Table 7.3: Average precision and recall of the Unet-Res-3DW network. Watermark retrieval from a single image, and by majority vote on ensembles of 20 images.

Dataset	Recall		Precision	
	Single	Ensemble	Single	Ensemble
Red	0.82	**0.89**	0.76	**0.90**
yellow	0.84	**0.91**	0.88	**0.94**
green	0.84	**0.92**	0.88	**0.98**
white	0.84	**0.99**	0.86	**1.00**
black	0.80	**0.92**	0.81	**0.91**
blue	0.81	**0.88**	0.80	**0.86**
purple	0.73	**0.83**	0.73	**0.84**
metal	0.74	**0.90**	0.83	**0.92**

network, which in some cases can exceed 15% in both the precision and the recall rates. Interestingly, both light probes and synthetic illumination models can give good results, as we can see from the illumination models 1 and 7, which are a light probe and a synthetic model, respectively, and they trained the two-best performing networks. We notice that in both cases there is a similarity between the scene corresponding to the illumination model and the environment in which the test set images were taken, that is, indoors well-lit scenes with strong ambient light coming from the windows. In contrast, the illumination model 2, which lacks ambient light, and illumination models 3 and 6, which correspond to outdoor environments, do not perform as well.

The illumination models 4 and 5 do not perform well, even though they are light probes from environments similar to the test environment, that is, indoor or covered outdoor scenes with strong ambient light. However, as we can see from Table 7.1, their resolution is low, and the rendered images lack crucial detail. We also notice that the illumination model 8 does not perform well, despite its close resemblance to the test environment. In fact, the illumination model 8, with morning skylight entering the room through windows, is technically more faithful to the test environment than illumination model 7, where the windows are area lights, i.e., light-emitting surfaces. Nevertheless, the complexity of the 3D scene in illumination model 8 means that various uncontrolled heavy shadows appear, see Figure 7.10,

Table 7.4: Precision and recall of Unet-Res-3DW in ensemble mode for each training/test set combination.

Dataset	Illum. 1		Illum. 2		Illum. 3		Illum. 4		Illum. 5		Illum. 6		Illum. 7		Illum. 8	
	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.
Red	1.00	1.00	0.81	0.82	0.90	0.97	0.96	0.91	0.92	0.86	1.00	1.00	1.00	1.00	0.88	0.83
Yellow	1.00	1.00	0.84	0.82	0.90	0.89	0.95	0.89	0.90	0.88	0.82	0.90	1.00	1.00	0.80	0.85
Green	0.99	1.00	0.82	0.83	0.85	0.81	0.89	0.84	0.82	0.80	0.84	0.98	0.98	0.95	0.76	0.70
White	1.00	1.00	0.89	0.90	0.93	0.95	0.96	0.94	0.94	0.91	1.00	1.00	1.00	0.99	0.83	0.86
Black	0.90	0.91	0.77	0.74	0.80	0.79	0.83	0.87	0.76	0.83	0.83	0.85	0.94	0.90	0.81	0.79
Blue	0.98	1.00	0.84	0.88	0.84	0.86	0.90	0.90	0.89	0.84	0.79	0.95	0.93	0.88	0.83	0.75
Purple	0.97	1.00	0.80	0.80	0.80	0.85	0.88	0.86	0.83	0.87	0.78	0.79	0.89	0.92	0.84	0.80
Metal	0.96	0.93	0.82	0.81	0.79	0.75	0.91	0.84	0.78	0.84	0.84	0.85	0.91	0.88	0.73	0.84
Avg.	**0.98**	**0.98**	0.82	0.83	0.85	0.86	0.91	0.88	0.86	0.85	0.86	0.92	**0.96**	**0.94**	0.81	0.80
St.d.	0.03	0.04	0.04	0.05	0.05	0.08	0.05	0.04	0.07	0.03	0.09	0.08	0.04	0.05	0.05	0.05

impacting the quality of the synthetic training set.

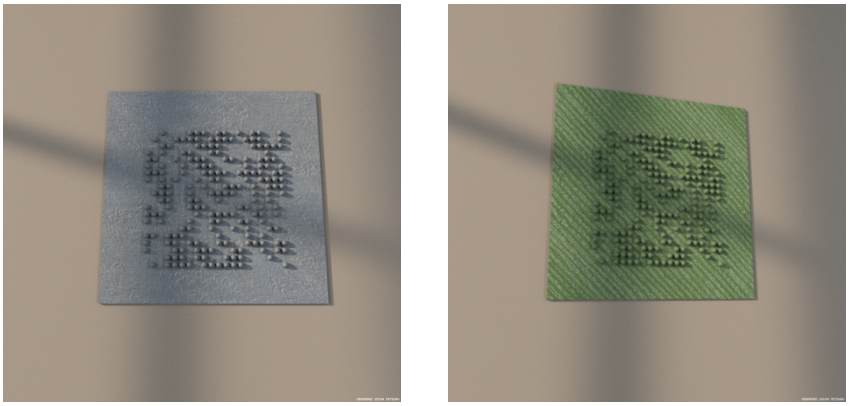


Figure 7.10: Training images generated with the illumination model 8.

7.4.3 Additional experiment

In this experiment, it is designed to explore when changing the illumination for a simple scene, if there are the similar results for detecting the watermarks.

Here, I still use the office room scene in figure 7.4, but only change the light intensity with 25%, 50%, 75% and 100%. When the light intensity with 75% is most close resemblance to the test environment. I generated four training datasets of 2K images each, one for each illumination model. The other parameters as same as above experiments setting.

Table 7.5: Precision and recall of Unet-Res-3DW in ensemble mode for different light intensity.

Dataset	25% light intensity		50% light intensity		75% light intensity		100% light intensity	
	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.	Rec.	Pr.
Red	0.77	0.75	0.85	0.82	1.00	1.00	0.86	0.81
Yellow	0.76	0.77	0.86	0.85	1.00	1.00	0.85	0.88
Green	0.77	0.75	0.82	0.83	0.98	0.95	0.87	0.84
White	0.80	0.78	0.84	0.90	1.00	0.99	0.90	0.84
Black	0.70	0.70	0.79	0.80	0.94	0.90	0.81	0.81
Blue	0.74	0.73	0.81	0.82	0.93	0.90	0.85	0.87
Purple	0.74	0.72	0.80	0.80	0.89	0.92	0.83	0.82
Metal	0.75	0.73	0.82	0.81	0.91	0.88	0.85	0.84
Avg.	0.75	0.74	0.82	0.83	0.96	0.94	0.85	0.84
St.d.	0.03	0.02	0.02	0.03	0.04	0.05	0.02	0.02

From this table 7.5, we can see that also provide the similar result that even with the same light source, when the light intensity is more close to the test environment, the predicted result is better. Obviously, the impact of the illumination is significant.

7.5 Conclusion

We studied the effect of the illumination model on the quality of synthetic training datasets for deep neural networks. The images of the training sets are renderings of planar surfaces with 20×20 bits arrays imprinted on them in the form semi-spherical bumps and arranged at the nodes of a regular grid. Several network/training set combinations were tested on images captured from 3D printings of these 3D models. The various training sets correspond to different illumination models, some of which are based on natural environments recorded in light probes and some of which are based on synthetic light modelling. Our main findings can be summarized as follows.

Appropriate choice of illumination model can lead to increases in the accuracy rates over 15%. However, good network performance can be achieved using either

captured natural light or synthetic illumination models. In both cases, the degree of similarity between the illumination model and the natural environment in which the test images are captured is important. In the case of captured natural light, the resolution of the light probe is also an important factor. Finally, uncontrolled artefacts from complex synthetic illumination models, and in particular shadows, can degrade network performance.

Our experiment shows that the impact of the illumination model is significant, and the potential gains from its optimization can easily exceed those from the optimization of the network architecture. However, we also note that the main findings of experiments could have either been predicted or could have been deduced by visual inspection of few representative images of the training set. In the future, we would like to establish relationships between the illumination model and classifier's performance that are neither obvious nor can be deduced by visual inspection of the training set. We expect these relationships to be subtler than the ones we have studied here, and thus, a more massive scale experiment would be needed to establish them.

Chapter 8

Conclusions and Future Works

In this thesis, we addressed challenges related to watermarking on 3D printed objects. We explored numerous parameters of the problem, including printing methods, colours, textures, illumination environments, camera viewpoint, quality and number of acquired images of the watermarked object. Because watermarking on 3D printed objects is a very nascent research area, there are no benchmark databases and relatively few existing algorithms. Generally, in studying the various parameters of the problem, we developed a deep learning based retrieval methods, which already perform acceptably well and seem can be improved further.

The various parameters of the process of watermarking 3D printed objects change the appearance of 3D printed watermarks in different ways. For example, choice of printing method may change the amount of noise on both the 3D printed watermarks and the background; camera viewpoint and illumination conditions alter the visual appearance of the captured images of the 3D printed objects; printing materials and printing textures may change the appearance of any part of the printed object. We viewed these effect caused by parameters of the process as a partial features corruption problem with unknown locations in a certain domain, claiming that a dynamic feature selection process could address these issues. For example, given a set of images of 3D printed watermarked objects with unknown parameters, only the relevant 3D printed watermark features should dynamically be chosen and used for segmentation. Nevertheless, it is difficult to measure such features early on the training stage, and even the recognised relevant features could become irrelevant due

to feature corruption caused by unknown covariates. We claim that a DNN-based segmentation enables us to address this issue, as the semantic segmentation with the irrelevant features can be eliminated at the stage of decision-level fusion.

We also imported synthetic data to augment the limited training data, and domain randomisation to reduce the gap between synthetic and real-world images data, obtaining significant performance improvement. After discussing the theoretical basis and providing proof-of-concept work for DNN-based semantic segmentation methods, we further proposed several novel DNN architectures, such as CNN-3DW and Unet-3DW. Moreover, we introduced various data augmentation methods, such as flip, crop, colour space transformations, noise injection, adversarial network generation, style transfer, and physically synthetic image renders. We further combined them with domain randomisation to solve the problem of accurate 3D watermark retrieval against backdrop challenges such as printing noise, variation in printing materials, or unfavourable camera viewpoint.

8.1 Conclusions

Summarising the achievements and the limitations of this thesis per chapter, in Chapter 3, we laid down the problem and introduced our research framework for studying 3D watermarking of printed 3D objects, before concentrating in subsequent chapters on view independent visible watermarks on a flat surface. In this chapter, we adapted a traditional image processing method, LBP, and used to for the segmentation of 3D watermarking bumps. Because this field is nascent, the LBP results also serve as a baseline for our subsequent research documented in later chapters. Although LBPs gave some encouraging results on some images, especially those taken under favourable illumination conditions and camera viewpoints, the overall experimental results suggest that significant improvements are needed to be able to retrieve information from more general images and for handling more significant intra-class variations. Finally, we also include some basic discussion and results on view-dependent watermarks, watermarks on the non-flat surface, and watermarks embedded under the object's surface.

In Chapter 4, to address the issue of large intra-class variations caused by the purely geometric nature of the information-carrying features, we employed state-of-the-art computer vision technologies to improve on the baseline algorithm which was based on traditional image processing. With the DNN architecture we employed, an amount of high-quality training data is crucial for a neural network to learn and acquire generalisation power. Due to the limited number of original training image data, we use random rotations, random cropping and random changes to the brightness, to augment our data and achieve robust training. In this chapter, we proposed a new DNN architecture, CNN-3DW, to retrieve watermark bumps. Our results showed that the CNN-3DW based computer vision techniques are capable of tackling effectively challenges related to the variability of parameters, such as camera view, illumination environment and printing noise. The CNN-3DW results compared favourably against the LBP method, but the relatively limited experimentation was not enough to prove CNN-3DW's ability to cope with the challenges of real application scenarios. Hence, more extensive experimentation had to be considered.

In Chapter 5, extending the work in Chapter 4, we added more experiments to prove the effectiveness of CNN-3DW in 3D printed watermarking bump segmentation. One of the directions was cross-material verification, enforcing the rule that the training and test sets should not contain images of objects printed with the same material. The other direction was the active learning method. In the active learning approach, we expanded the training set with examples on which the network underperformed, and then retrained. We noticed that the results improved significantly, and concluded that in our setting the choice of suitable images for inclusion in the training data is really important. Furthermore, we extended the CNN-3DW method to other watermark carrier surfaces, such as non-planar ones, showing that the network is still useful. However, we also established various limitations of this network, such as the requirement for a large number of training data, the reliance on corner landmarks for identifying the RoI, and perhaps, more importantly, some roughness in the reconstructed confidence maps.

In Chapter 6, addressing the limitation that we need more high-quality training

data to produce fine confidence maps, we used synthetic image data to augment the real training data, and domain randomisation to build the relation between synthetic data and real-world data. By using synthetic DR data, the neural network was forced to focus on the most relevant features between the synthetic watermark bumps and the real-world ones. We showed that by mixing synthetic DR data and real image data, the retrieval rate is higher than when using real only or synthetic only data. Moreover, by performing a systematic ablation study, we showed that the training dataset size, the material of the printed 3D objects and the illumination environment, all affect the retrieval accuracy of the 3D watermarking bumps.

In Chapter 7, we presented a systematic study of the effect of the illumination model on the quality of the DNN synthetic training data set. Several network/synthetic training set combinations were tested on images captured from 3D printed of the 3D models that were used to render the synthetic images. The various training sets corresponded to different lighting models, some of which were natural illumination environments captured in light probes. In contrast, some other synthetic training sets were rendered using synthetic light modelling. Our experiments showed that the impact of the lighting model is very significant, and the potential benefits of its optimisation can easily exceed the benefits of network architecture optimisation. However, we also note that the main findings of the experiment were rather expected and could have been predicted in advance, or could have been inferred by visual inspection of several representative images of the training sets.

8.2 Future works

Although, as I have just said, we consider that the goal of the thesis has been accomplished, there are plenty of improvements that could be done in order to achieve better results.

In the future, I would like to explore other direction of 3D printed watermarking, such as view independent 3D printed watermarking under the surfaces and view dependent 3D printed watermarking, which is explained in chapter 3.

For the information retrieval process, I can also improve the algorithm and subse-

quently improve performance. It is worth strengthening the merge stage to increase robustness and reduce the possibility of insufficient segmentation. Another potentially beneficial extension is to modify the information retrieval algorithm to perform recursive, coarse-to-fine segmentation to achieve better performance and allows the algorithm to segment higher resolution images.

Besides, I hope to establish relationships between the illumination model and classifier's performance that are neither obvious nor can be deduced by visual inspection of the training set. We expect these relationships to be subtler than the ones I have studied here, and thus, a more massive scale experiment would be needed to establish them.

Bibliography

[1] Gibson, I., Rosen, D. W. and Stucker, B. (2014). *Additive manufacturing technologies.* Vol. 17. New York: Springer.

[2] Ngo, T. D., Kashani, A., Imbalzano, G., Nguyen, K. T. and Hui, D. (2018). Additive manufacturing (3D printing): A review of materials, methods, applications and challenges. *Composites Part B: Engineering*, 143, 172-196.

[3] PlasticsToday Staff in Medical, (2020), *Formlabs Is 3D Printing More than 100,000 COVID-19 Test Swabs Daily.*, Available at:https://www.plasticstoday.com/medical/formlabs-3d-printing-more-100000-covid-19-test-swabs-daily/38291431262722 [Accessed 18th April 2020]

[4] Stratasys, (2020), *CoVent-19 Challenge Asks Millions of Designers and Engineers to Respond to the Ventilator Crisis.*, Available at:https://investors.stratasys.com/news-events/press-releases/detail/527/covent-19-challenge-asks-millions-of-designers-and [Accessed 18th April 2020]

[5] Sher, D. (2020),*MIT emergency ventilator E-Vent project is ready.* Available at:https://www.3dprintingmedia.network/mit-emergency-ventilator-e-vent-project-is-ready/ [Accessed 18th April 2020]

[6] Ślusarczyk, P. (2020). *The personal door opener – to keep your hands free of coronavirus risk.* Available at:https://3dprintingcenter.net/(2020)/03/30/the-personal-door-opener-to-keep-your-hands-free-of-coronavirus-risk/ [Accessed 18th April 2020]

[7] Czech Institute of Informatics Robotic and Cybernetics. (2020). *CIIRC CTU Develops Own Prototype of CIIRC RP95 Respirator / Half Mask.* Available at: https://www.ciirc.cvut.cz/covid-2/ [Accessed 18th April 2020]

[8] MARINE CORPS LANCE CPL. GERARDO CANO. (2020), April. *Marines Put 3D Printing to Use in COVID-19 Fight.* Available at: https://www.defense.gov/Explore/Features/Story/Article/2142404/marines-put-3d-printing-to-use-in-covid-19-fight/ [Accessed 18th April 2020]

[9] Wohlers, T. and Gornet, T. (2014). History of additive manufacturing. *Wohlers report, 24*2014, 118.

[10] Stazzone, S. (2020). *Guide to Barcode Types and Standards: 1D, 2D Barcode Symbologies, Requirements, and Standards-Issuing Entities.* Available at: https://www.camcode.com/asset-tags/guide-to-barcode-types-standards [Accessed 13rd February 2020]

[11] Soon, T. J. (2008). QR code. *Synthesis Journal, 2008* , 59-78.

[12] Plain-Jones, C. (1995). Data matrix identification. *Sensor Review, 15*(1), 12-15.

[13] Nagasaki, T., Fujimori, H., Mori, T., Matsui, S., Akamine, Y., Wakamatsu, S. and Morita, K. (1999). *U.S. Patent No. 5,896,403.* Washington, DC: U.S. Patent and Trademark Office.

[14] Acumen Research and Consulting. (2020). *3D Printing Market By Technology (Stereo Lithography, Fused Deposition Modelling, Selective Laser Sintering, Electron Beam Melting, Digital Light Processing, Other Technologies), By Offering (Printer, Material, Software, Service), By End User (Automotive, Healthcare, Industrial, Consumer electronics, Aerospace and defense, Others)– Global Industry Analysis, Market Size, Opportunities And Forecast, 2019-2026.* Available at: https://www.acumenresearchandconsulting.com/3d-printing-market [Accessed: February 13rd 2020].

[15] Ratto, M. and Ree, R. (2012). Materializing information: 3D printing and social change. *First Monday*. 17(7).

[16] Winter, J. (2013). *Homeland Security bulletin warns 3D-printed guns may be 'impossible' to stop*. Available at: https://www.foxnews.com/us/homeland-security-bulletin-warns-3d-printed-guns-may-be-impossible-to-stop [Accessed 15th February 2020].

[17] Formlabs. (2020). *FDM vs. SLA: 2020 3D Printer Comparison Guide.* Available at: https://formlabs.com/blog/fdm-vs-sla-compare-types-of-3d-printers/ [Accessed on 16th February 2020]

[18] Schoffer, F. (2016) *How expiring patents are ushering in the next generation of 3D printing.* Available at: https://techcrunch.com/2016/05/15/how-expiring-patents-are-ushering-in-the-next-generation-of-3d-printing/ [Accessed on 16th February 2020]

[19] Filabot. (n.d.) Available at: https://www.filabot.com/ [Accessed on 17th February 2020]

[20] Armstrong, C. (n.d.) *Post processing for FDM printed parts.* Available at: https://www.3dhubs.com/knowledge-base/post-processing-fdm-printed-parts/#sanding [Accessed on 17th February 2020]

[21] Treatstock. (2020). Available at: https://www.treatstock.com/site/about [Accessed on 5th March 2020]

[22] Watermark. (n.d.) Available at: https://www.watermark3d.com/ [Accessed on 5th March 2020]

[23] Price, K. (2018). *SLAC Scientists Investigate How Metal 3-D Printing Can Avoid Producing Flawed Parts.* Available at: https://www6.slac.stanford.edu/news/2018-01-30-slac-scientists-investigate-how-metal-3-d-printing-can-avoid-producing-flawed-parts [Accessed on 5th March 2020]

[24] Garcia-Lamont, F., Cervantes, J., López, A., and Rodriguez, L. (2018). Segmentation of images by color features: A survey. *Neurocomputing*, 292, 1-27.

[25] Sacco, M. (1984). Stochastic Relaxation, Gibbs Distributions and Bayesian Restoration of Images. *IEEE Trans Pattern Anal Mach Intell*6:721-741

[26] Arbelaez, P., Maire, M., Fowlkes, C. and Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence, 33*(5), 898-916.

[27] Zhang, H., Sindagi, V. and Patel, V. M. (2019). Image de-raining using a conditional generative adversarial network. *IEEE transactions on circuits and systems for video technology.*

[28] Wang, K., Lavoué, G., Denis, F. and Baskurt, A., (2007), Three-dimensional meshes watermarking: Review and attack-centric investigation. In*International Workshop on Information Hiding* (pp. 50-64). Springer, Berlin, Heidelberg.

[29] Zhu, J.Y., Wu, J., Xu, Y., Chang, E. and Tu, Z. (2014). Unsupervised object class discovery via saliency-guided multiple class learning. *IEEE transactions on pattern analysis and machine intelligence, 37*(4),862-875.

[30] Cheng, G., Han, J. and Lu, X. (2017). Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE, 105*(10), 1865-1883.

[31] Tian, Z., Shen, C., Chen, H. and He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 9627-9636.

[32] Liang, M., Yang, B., Chen, Y., Hu, R. and Urtasun, R. (2019). Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7345-7353.

[33] Li, S., Seybold, B., Vorobyov, A., Fathi, A., Huang, Q. and Jay Kuo, C.C. (2018). Instance embedding transfer to unsupervised video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6526-6535.

[34] Wang, W., Song, H., Zhao, S., Shen, J., Zhao, S., Hoi, S.C. and Ling, H. (2019). Learning unsupervised video object segmentation through visual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3064-3074.

[35] Lu, X., Wang, W., Ma, C., Shen, J., Shao, L. and Porikli, F. (2019). See more, know more: Unsupervised video object segmentation with co-attention siamese networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3623-3632.

[36] Xian, Y., Choudhury, S., He, Y., Schiele, B. and Akata, Z. (2019). Semantic projection network for zero-and few-label semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* , 8256-8265.

[37] Fu, J., Liu, J., Wang, Y., Zhou, J., Wang, C. and Lu, H. (2019). Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing.*

[38] Ojala, T., Pietikainen, M. and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence, 24*(7), 971-987.

[39] Milan, A., Rezatofighi, S.H., Dick, A., Reid, I. and Schindler, K. (2017). Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence.*

[40] Bouwmans, T., Silva, C., Marghes, C., Zitouni, M.S., Bhaskar, H. and Frelicot, C. (2018). On the role and the importance of features for background modeling and foreground detection. *Computer Science Review, 28*, 26-91.

[41] Shi, Z., Yu, X., Jiang, Z., and Li, B. (2013). Ship detection in high-resolution optical imagery based on anomaly detector and local shape feature. *IEEE Transactions on Geoscience and Remote Sensing, 52*(8), 4511-4523.

[42] Ma, B., Liu, Z., Jiang, F., Yan, Y., Yuan, J. and Bu, S. (2019). Vehicle detection in aerial images using rotation-invariant cascaded forest. *IEEE Access, 7*, 59613-59623.

[43] Guo, Z., Zhang, L. and Zhang, D. (2010). A completed modeling of local binary pattern operator for texture classification. *IEEE transactions on image processing, 19*(6), 1657-1663.

[44] Han, J., Zhang, D., Cheng, G., Guo, L. and Ren, J. (2014). Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning. *IEEE Transactions on Geoscience and Remote Sensing, 53*(6), 3325-3337.

[45] Li, K., Wan, G., Cheng, G., Meng, L. and Han, J. (2020). Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing, 159*, 296-307.

[46] Pietikäinen, M. and Zhao, G. (2015). Two decades of local binary patterns: A survey. In *Advances in independent component analysis and learning machines* (pp. 175-210). Academic Press.

[47] Brahnam, S., Jain, L. C., Nanni, L. and Lumini, A. (Eds.). (2014). *Local binary patterns: new variants and applications* (Vol. 2). Berlin: Springer.

[48] Singhal, S. and Tripathi, V. (2019). Action recognition framework based on normalized local binary pattern. In *Progress in Advanced Computing and Intelligent Engineering*, 247-255.

[49] Ojala, T., Pietikainen, M. and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence, 24*(7), 971-987.

[50] Liu, L., Lao, S., Fieguth, P.W., Guo, Y., Wang, X. and Pietikäinen, M. (2016). Median robust extended local binary pattern for texture classification. *IEEE Transactions on Image Processing, 25*(3), pp.1368-1381.

[51] Tan, X. and Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE transactions on image processing, 19*(6), 1635-1650.

[52] Yang, W. and Sun, C. (2011, June). Face recognition using improved local texture patterns. In *2011 9th World Congress on Intelligent Control and Automation* (pp. 48-51). IEEE.

[53] Akhloufi, M.A. and Bendada, A. (2010, October). Locally adaptive texture features for multispectral face recognition. In *2010 IEEE International Conference on Systems, Man and Cybernetics* (pp. 3308-3314). IEEE.

[54] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[55] Papandreou, G., Chen, L.C., Murphy, K.P. and Yuille, A.L. (2015). Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1742-1750).

[56] Hong, S., Noh, H. and Han, B. (2015). Decoupled deep neural network for semi-supervised semantic segmentation. In *Advances in neural information processing systems* (pp. 1495-1503).

[57] Noh, H., Hong, S. and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1520-1528).

[58] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv*:1412.7062.

[59] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence, 40*(4), pp.834-848.

[60] Jégou, S., Drozdzal, M., Vazquez, D., Romero, A. and Bengio, Y. (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 11-19).

[61] Liu, W., Rabinovich, A. and Berg, A.C. (2015). Parsenet: Looking wider to see better. *arXiv preprint arXiv:*1506.04579.

[62] Ren, S., He, K., Girshick, R. and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In*Advances in neural information processing systems* (pp. 91-99).

[63] Dai, J., Li, Y., He, K. and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (pp. 379-387).

[64] Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision* (pp. 2650-2658).

[65] Long, J., Shelhamer, E. and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *IEEE Annals of the History of Computing,*(04), (pp.3431-3440)

[66] He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[67] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv*:1409.1556.

[68] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[69] Zhao, H., Shi, J., Qi, X., Wang, X. and Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2881-2890).

[70] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:*1511.07122.

[71] Lin, G., Shen, C., Van Den Hengel, A. and Reid, I. (2017). Exploring context with deep structured models for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence, 40*(6), pp.1352-1366.

[72] Ronneberger, O., Fischer, P. and Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*(pp. 234-241). Springer, Cham.

[73] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*(pp. 2117-2125).

[74] Wu, H., Zhang, J., Huang, K., Liang, K. and Yu, Y. (2019). FastFCN: Rethinking dilated convolution in the backbone for semantic segmentation. *arXiv preprint arXiv*:1903.11816.

[75] Takikawa, T., Acuna, D., Jampani, V. and Fidler, S. (2019). Gated-scnn: Gated shape cnns for semantic segmentation. In*Proceedings of the IEEE International Conference on Computer Vision*(pp. 5229-5238).

[76] Badrinarayanan, V., Kendall, A. and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence, 39*(12), pp.2481-2495.

[77] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*(pp. 4700-4708).

[78] Chen, L.C., Papandreou, G., Schroff, F. and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv*:1706.05587.

[79] Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818).

[80] Alvarez, J.M., LeCun, Y., Gevers, T. and Lopez, A.M. (2012, October). Semantic road segmentation via multi-scale ensembles of learned features. In *European Conference on Computer Vision* (pp. 586-595). Springer, Berlin, Heidelberg.

[81] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

[82] Nguyen, V., Yago Vicente, T. F., Zhao, M., Hoai, M. and Samaras, D. (2017). Shadow detection with conditional generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 4510-4518).

[83] Duan, Y., Lu, J., Feng, J. and Zhou, J. (2017). Learning rotation-invariant local binary descriptor. *IEEE Transactions on Image Processing, 26*(8), 3636-3651.

[84] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... and Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision, 115*(3), 211-252.

[85] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

[86] Shorten, C. and Khoshgoftaar, T.M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6*(1), 60.

[87] Moreno-Barea, F. J., Strazzera, F., Jerez, J. M., Urda, D. and Franco, L. (2018). Forward Noise Adjustment Scheme for Data Augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 728-734).

[88] DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv*:1708.04552.

[89] Goodfellow, I.J., Shlens, J. and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv*:1412.6572.

[90] Miyato, T., Maeda, S.I., Koyama, M. and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence, 41*(8), 1979-1993.

[91] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv*:1411.1784.

[92] Odena, A., Olah, C. and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 2642-2651).

[93] Zhang, X., Wang, Z., Liu, D. and Ling, Q. (2019). Dada: Deep adversarial data augmentation for extremely low data regime classification. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2807-2811). IEEE.

[94] Jackson, P. T., Atapour-Abarghouei, A., Bonner, S., Breckon, T. P. and Obara, B. (2019). Style Augmentation: Data Augmentation via Style Ran-

domization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 83-92).

[95] Such, F.P., Rawal, A., Lehman, J., Stanley, K .O. and Clune, J. (2019). Generative Teaching Networks: Accelerating Neural Architecture Search by Learning to Generate Synthetic Training Data. *arXiv preprint arXiv*:1912.07768.

[96] Antoniou, A., Storkey, A. and Edwards, H. (2018). Augmenting image classifiers using data augmentation generative adversarial networks. In *International Conference on Artificial Neural Networks* (pp. 594-603). Springer, Cham.

[97] Chongxuan, L.I., Xu, T., Zhu, J. and Zhang, B. (2017). Triple generative adversarial nets. In *Advances in neural information processing systems*(pp. 4088-4098).

[98] Tran, T., Pham, T., Carneiro, G., Palmer, L. and Reid, I. (2017). A bayesian data augmentation approach for learning deep models. In *Advances in neural information processing systems*(pp. 2797-2806).

[99] Gatys, L.A., Ecker, A.S. and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*(pp. 2414-2423).

[100] Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q.V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*(pp. 113-123).

[101] Zhang, Q., Yang, L. T., Chen, Z., and Li, P. (2018). A survey on deep learning for big data. *Information Fusion, 42*, 146-157.

[102] Pytorch. (2020). *TORCH.UTILS.DATA*. Available at: https://pytorch.org/docs/stable/data.html [Accessed 27th July 2020]

[103] Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

[104] ImageNet, (2016), *ImageNet.* Available at:http://image-net.org/index [Accessed 27th July 2020]

[105] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... and Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision, 115*(3), 211-252.

[106] Liu, J., Shen, H., Narman, H.S., Chung, W. and Lin, Z. (2018). A survey of mobile crowdsensing techniques: A critical component for the internet of things. *ACM Transactions on Cyber-Physical Systems, 2*(3), 1-26.

[107] Daniel, F., Kucherbaev, P., Cappiello, C., Benatallah, B. and Allahbakhsh, M. (2018). Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR), 51*(1), 1-40.

[108] Sun, B. and Saenko, K. (2014), September. From Virtual to Reality: Fast Adaptation of Virtual Object Detectors to Real Domains. In*The British Machine Vision Conference*(Vol.1, No.2, p.3).

[109] Hinterstoisser, S., Lepetit, V., Wohlhart, P., and Konolige, K. (2018). On pre-trained image features and synthetic images for deep learning. In*Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 0-0).

[110] Movshovitz-Attias, Y., Kanade, T. and Sheikh, Y. (2016). How useful is photo-realistic rendering for visual learning?. In *European Conference on Computer Vision*(pp. 202-217). Springer, Cham.

[111] Hodaň, T., Vineet, V., Gal, R., Shalev, E., Hanzelka, J., Connell, T., ... and Guenter, B. (2019), September. Photorealistic image synthesis for object instance detection. In *2019 IEEE International Conference on Image Processing (ICIP)* (pp. 66-70). IEEE.

[112] Richter, S.R., Vineet, V., Roth, S. and Koltun, V. (2016). Playing for data: Ground truth from computer games. In*European conference on computer vision*(pp. 102-118). Springer, Cham.

[113] Cambridge University. (2007).*CamSeq01 Dataset, Cambridge Labeled Objects in Video.* Available at: http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamSeq01/ [Accessed 4th August 2020]

[114] Tremblay, K. (2019). *DATA MATRIX CODES VS. QR CODES – WHAT IS THE DIFFERENCE?*, Available at: https://www.laserax.com/blog/data-matrix-vs-qr-codes [Accessed 4th August 2020]

[115] Shafaei, A., Little, J.J. and Schmidt, M. (2016). Play and learn: Using video games to train computer vision models. *arXiv preprint arXiv*:1608.01745.

[116] Peng, X., Sun, B., Ali, K. and Saenko, K. (2015). Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1278-1286).

[117] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I. and Schmid, C. (2017). Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 109-117).

[118] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G. and Black, M. (2015). SMPL: a skinned multi-person linear model. In: *SIGGRAPH*

[119] Gao, J. Z., Prakash, L., and Jagatesan, R. (2007). Understanding 2d-barcode technology and applications in m-commerce-design and implementation of a 2d barcode processing solution, In*31st Annual International Computer Software and Applications Conference (COMPSAC 2007),IEEE* Vol.2, pp.49-56.

[120] The Global Language of Business. (2018). *GS1 DataMatrix Guideline, Overview and technical introduction to the use of GS1 DataMatrix.*, Release 2.5.1, GS1, Available at : https://www.gs1.org/docs/barcodes/GS1_DataMatrix_Guideline.pdf [Access 19 February 2020].

[121] The Global Language of Business. (2020). *GS1 General Specifications : The foundational GS1 standard that defines how identi-*

fication keys, data attributes and barcodes must be used in business applications., Release 20.0,GS1, pp.305-311, Available at : https://www.gs1.org/sites/default/files/docs/barcodes/GS1_General_Specifications.pdf [Access 19 February 2020].

[122] AIM Global. (2017). *Information technology — Automatic identification and data capture techniques — Bar code symbology specification—DotCode.*, Cranberry Township, PA:AIM,Inc.

[123] Wright, G. IV. (2019). *Revised 2D barcode offers potential in beverage, distribution, eCommerce markets.*, Available at: https://www.packagingdigest.com/bar-coding/revised-2d-barcode-offers-potential-in-beverage-distribution-ecommerce-markets-2019-07-31 [Accessed 20th February 2020]

[124] (1991). DOTCODES: THE IDEAL IDENTIFICATION SYSTEM FOR EXACTING APPLICATIONS. *Sensor Review*, Vol.11 No.1,pp 15-16, Available at: https://doi.org/10.1108/eb007835

[125] Berisso, K. (2018). DotCode Damage Testing, *Journal of Computer Sciences and Applications.*, Vol.6, No.1, 44, Available at: DOI:10.12691/jcsa-6-1-6 [Access: 21st February 2020].

[126] Finkenzeller, K. (2003). *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication.* John wiley and sons.

[127] Shim, I. Y. and Kim,S. W. (2016). Using circular dot pattern code tag for medical information on the round type medical package. *International Journal of Advanced Media and Communication,6*(1), 48-56

[128] Meggs, P. B. (1998). *A History of Graphic Design*(Third ed.). Jhon Wiley and Sons, Inc. p.58.ISBN 978-0-471-29198-5.

[129] Singh, P. and Chadha, R. S. (2013). A survey of digital watermarking techniques, applications and attacks. *International Journal of Engineering and Innovative Technology (IJEIT), 2*(9), pp 165-175.

[130] Dekel, T., Rubinstein, M., Liu, C. and Freeman, W.T. (2017). On the effectiveness of visible watermarks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*(pp. 2146-2154)

[131] Agarwal, N., Singh, A.K. and Singh, P.K. (2019). Survey of robust and imperceptible watermarking. *Multimedia Tools and Applications, 78*(7), pp.8603-8633.

[132] Feig, E. and Winograd, S. (1992). Fast algorithms for the discrete cosine transform.*IEEE Transactions on Signal processing, 40*(9), pp.2174-2193.

[133] Ganic, E. and Eskicioglu, A.M. (2005). Robust embedding of visual watermarks using discrete wavelet transform and singular value decomposition. *Journal of electronic imaging, 14*(4), p.043004.

[134] Kitamura, I., Kanai, S. and Kishinami, T. (2001). Copyright protection of vector map using digital watermarking method based on discrete Fourier transform. In*IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No. 01CH37217)* (Vol. 3, pp. 1191-1193). IEEE.

[135] Stanescu, D., Stratulat, M., Ciubotaru, B., Chiciudean, D., Cioarga, R. and Borca, D. (2007). Digital watermarking using Karhunen-Loeve transform. In *2007 4th International Symposium on Applied Computational Intelligence and Informatics* (pp. 187-190). IEEE.

[136] Tuohey, J. (2004). *Government Uses Color Laser Printer Technology to Track Documents.* Available at: https://www.pcworld.com/article/118664/article.html [Accessed 4th May, 2020]

[137] Chalmers., B. (2015). *PRINTER STEGANOGRAPHY.* Available at: https://blog.findmysupplies.co.uk/printer-steganography/ [Accessed from 4th May 2020]

[138] Qu, Z. and Zhang, L. (2010), August. Research on image segmentation based on the improved Otsu algorithm. In 2010 *Second International Conference on Intelligent Human-Machine Systems and Cybernetics* (Vol.2, pp.228-231). IEEE.

[139] Zhu, N., Wang, G., Yang, G. and Dai, W. (2009). A fast 2d otsu thresholding algorithm based on improved histogram. In*2009 Chinese Conference on Pattern Recognition* (pp.1-5). IEEE.

[140] Liu, D. and Yu, J. (2009) August. Otsu method and K-means. In *2009 Ninth International Conference on Hybrid Intelligent Systems* (Vol. 1, pp. 344-349). IEEE.

[141] MathWorks. (2020). *Approaches to Registering Images.* Available at: https://uk.mathworks.com/help/images/approaches-to-registering-images.html [Accessed 10th May 2020]

[142] Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* (6), 583-598.

[143] Shapiro, L. and Stockman, G. (2020). *Computer vision.* Prentice Hall. pp.69-73

[144] Thort, S. (n.d.) *Best furniture design software — Free online ,Open Source.* Available at: https://learnmech.com/best-furniture-design-software-free-online-open-source/ [Accessed 14th May 2020]

[145] OpenScad. (n.d.) *OpenSCAD The programmers Solid 3D CAD Modeller.* Available at: http://www.openscad.org/documentation.html [Accessed 15th May 2020]

[146] Dey, D., (2017). *6 Feature-rich OpenSource CAD Tools for Linux*. Available at: https://linoxide.com/linux-how-to/cad-tools-linux/ [Accessed 4th August 2020]

[147] Macq, B., Alface, P.R. and Montanola, M. (2015). Applicability of watermarking for intellectual property rights protection in a 3D printing scenario. In *Proceedings of the 20th International Conference on 3D Web Technology* (pp. 89-95).

[148] Hou, J.U., Kim, D.G., Choi, S. and Lee, H.K. (2015). 3D print-scan resilient watermarking using a histogram-based circular shift coding structure. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security* (pp. 115-121).

[149] Zeiler, M.D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q.V., ... and Hinton, G.E. (2013). On rectified linear units for speech processing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*(pp. 3517-3521). IEEE.

[150] Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization.*arXiv preprint arXiv*:1412.6980.

[151] LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.

[152] Ketkar, N. (2017). Introduction to keras. In *Deep learning with Python* (pp. 97-111). Apress, Berkeley, CA.

[153] Zhang, X., Wang, Q., Breckon, T. and Ivrissimtzis, I. (2018). Watermark Retrieval from 3D Printed Objects via Convolutional Neural Networks. *arXiv preprint arXiv:*1811.07640.

[154] Zhang, Y., Qiu, W., Chen, Q., Hu, X. and Yuille, A. (2016), Unrealstereo: A synthetic dataset for analyzing stereo vision. *arXiv preprint arXiv:*1612.04647.

[155] Qiu, W. and Yuille, A. (2016). Unrealcv: Connecting computer vision to unreal engine. In*European Conference on Computer Vision* (pp. 909-916). Springer, Cham.

[156] Barron, J.L., Fleet, D. J. and Beauchemin, S.S. (1994). Performance of optical flow techniques. *International journal of computer vision, 12*(1), 43-77.

[157] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. and Abbeel, P. (2017, September). Domain randomization for transferring deep neural networks from simulation to the real world. In*2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*(pp. 23-30). IEEE.

[158] Jaderberg, M., Simonyan, K., Vedaldi, A. and Zisserman, A. (2014). Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv*:1406.2227.

[159] Gupta, A., Vedaldi, A. and Zisserman, A. (2016). Synthetic data for text localisation in natural images. In*Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2315-2324).

[160] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., ... and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2758-2766).

[161] Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S. and Cipolla, R. (2016). Understanding real world indoor scenes with synthetic data. In*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4077-4085).

[162] Atapour-Abarghouei, A. and Breckon, T.P. (2018). Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2800-2810).

[163] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., ... and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 969-977).

[164] Pov-Ray. (2018). *POV-Ray Persistence of Vision Raytracer*. Available at: http://www.povray.org/ [07.10.2018]

[165] Brown, L.G. (1992). A survey of image registration techniques. *ACM computing surveys (CSUR), 24*(4), 325-376.

[166] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics, 9*(1), 62-66.

[167] Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. (2014). How transferable are features in deep neural networks?. In*Advances in neural information processing systems* (pp. 3320-3328).

[168] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., and Berg, A.C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[169] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[170] He, K., Gkioxari, G., Dollár, P. and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

[171] Dai, J., He, K., Li, Y., Ren, S. and Sun, J. (2016). Instance-sensitive fully convolutional networks. In *European Conference on Computer Vision* (pp. 534-549). Springer, Cham.

[172] Li, Y., Qi, H., Dai, J., Ji, X. and Wei, Y. (2017). Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2359-2367).

[173] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

[174] Dai, J., Li, Y., He, K. and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In*Advances in neural information processing systems* (pp. 379-387).

[175] Wohlers, T.T. and Caffrey, T. (2015). *Wohlers report 2015: 3D printing and additive manufacturing state of the industry annual worldwide progress report.* Wohlers Associates.

[176] He, K., Zhang, X., Ren, S. and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence, 37*(9), 1904-1916.

[177] Ronneberger, O., Fischer, P. and Brox, T. (2015), October. U-net: Convolutional networks for biomedical image segmentation. In*International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

[178] He, Y., Shi, J., Wang, C., Huang, H., Liu, J., Li, G., ... and Wang, J. (2019). Semi-supervised Skin Detection by Network with Mutual Guidance. In*Proceedings of the IEEE International Conference on Computer Vision* (pp. 2111-2120).

[179] mitsuba-renderer . (2019). *Mitsuba Physically Based Render*. Available at: https://www.mitsuba-renderer.org/ [Access: 21st February 2020].

[180] Paul Debevec. (2004). *High-Resolution Light Probe Image Gallery*. Available at: https://people.ict.usc.edu/ debevec/Probes/ [Access: 21st February 2020].

[181] Bernhard Vogl. (2019). *Bernhard Vogl Light probes*. Available at: http://dativ.at/lightprobes/ [Access: 21st February 2020].

[182] Preetham, A.J., Shirley, P. and Smits, B. (1999). A practical analytic model for daylight. In*Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (pp. 91-100).

[183] Shubham, J. (2018). *What exactly does CNN see?*. Available at: https://becominghuman.ai/what-exactly-does-cnn-see-4d436d8e6e52 [Access: 20th February 2020].

[184] Zhang, Y., Zhou, D., Chen, S., Gao, S., and Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 589-597).

[185] Developable surface, (2021), *Wikipedia.*, Available at: https://en.wikipedia.org/wiki/Developable_surface [Accessed 18th February 2021]

[186] Rubens, N., Elahi, M., Sugiyama, M. and Kaplan, D., (2015). Active learning in recommender systems. In *Recommender systems handbook* (pp. 809-846). Springer, Boston, MA.

[187] Pinshape, (2021), *Pinshape.*, Available at: https://pinshape.com/ [Accessed 18th February 2021]

[188] MakerBot, (2021), *Thingiverse*, Available at: https://www.thingiverse.com/ [Accessed 18th February 2021]

[189] MyMiniFactory, (2021), *MyMiniFactory*, Available at:https://www.myminifactory.com/ [Accessed 18th February 2021]

[190] Dietterich, T.G., (2002), Ensemble learning. *The handbook of brain theory and neural networks*, 2, pp.110-125.

[191] MathWorks, (2021), *regionprops*, Available at: https://www.mathworks.com/help/images/ref/regionprops.html [Accessed 18th February]

[192] MathWorks, (2021),*Register Images with Projection Distortion Using Control Points* Available at: https://www.mathworks.com/help/images/registering-an-aerial-photo-to-an-orthophoto.html [Accessed 18th February]