# On Motion Analysis in Computer Vision with Deep Learning: Selected Case Studies

**By**

**Essa Anas**

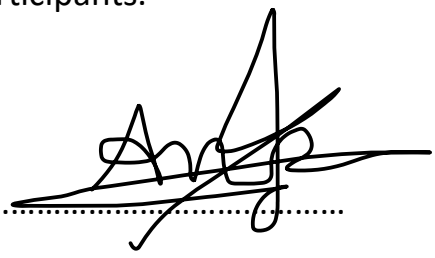
**School of Engineering**

A thesis submitted for the degree of Doctor of Philosophy at the University of Central Lancashire

December 2020

# Declaration

I hereby declare that this thesis represents my own work, which has been done after registration for the degree of PhD as appropriate in the school Engineering, University of Central Lancashire, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications. I have read the University's current research ethics guidelines and accept responsibility for the conduct of the procedures in accordance with the University's directions. I have attempted to identify all the risks related to this research that may arise in conducting this research, obtained the relevant ethical and/or safety approval (where applicable), and acknowledged my obligations and the rights of the participants.



Signature……

Date……14/12/2020……

# Acknowledgment

I wish to express my sincere appreciation to the JOST institute in School of Engineering and its president Prof. Ian Sherrington for offering the necessary budget to cover this Ph.D. study. Sincere Appreciation also to the supervisory team for their efforts in making this project see the like and specially to my supervisor, Prof. Bogdan Matuszewski, for his continuous guidance and encouragement and his professional support even when the road getting tight. With his experience, the goal of this project would not have been realized. The same gratitude and thanks acknowledged to the rest of supervisory team for their continuous encouragement and support that I had which enhance my study.

I wish to acknowledge the support and great love of my family. They kept me going on and this work would not have been possible without their patience.

# Abstract

Motion analysis is one of the essential enabling technologies in computer vision. Despite recent significant advances, image-based motion analysis remains a very challenging problem. This challenge arises because the motion features are extracted directory from a sequence of images without any other meta data information. Extracting motion information (features) is inherently more difficult than in other computer vision disciplines.

In a traditional approach, the motion analysis is often formulated as an optimisation problem, with the motion model being hand-crafted to reflect our understanding of the problem domain. The critical element of these traditional methods is a prior assumption about the model of motion believed to represent a specific problem. Data analytics' recent trend is to replace hand-crafted prior assumptions with a model learned directly from observational data with no, or very limited, prior assumptions about that model. Although known for a long time, these approaches, based on machine learning, have been shown competitive only very recently due to advances in the so-called deep learning methodologies.

This work's key aim has been to investigate novel approaches, utilising the deep learning methodologies, for motion analysis where the motion model is learned directly from observed data. These new approaches have focused on investigating the deep network architectures suitable for the effective extraction of spatiotemporal information. Due to the estimated motion parameters' volume and structure, it is frequently difficult or even impossible to obtain relevant ground truth data. Missing ground truth leads to choose the unsupervised learning methodologies which is usually represents challenging choice to utilize in already challenging high dimensional motion representation of the image sequence. The main challenge with unsupervised learning is to evaluate if the algorithm can learn the data model directly from the data only without any prior knowledge presented to the deep learning model during

In this project, an emphasis has been put on the unsupervised learning approaches. Owning to a broad spectrum of computer vision problems and applications related to motion analysis, the research reported in the thesis has focused on three specific motion analysis challenges and corresponding practical case studies. These include motion detection and recognition, as well as 2D and 3D motion field estimation.

Eyeblinks quantification has been used as a case study for the motion detection and recognition problem. The approach proposed for this problem consists of a novel network architecture processing weakly corresponded images in an action completion regime with learned spatiotemporal image features fused using cascaded recurrent networks.

The stereo-vision disparity estimation task has been selected as a case study for the 2D motion field estimation problem. The proposed method directly estimates occlusion maps using novel convolutional neural network architecture that is trained with a custom-designed loss function in an unsupervised manner.

The volumetric data registration task has been chosen as a case study for the 3D motion field estimation problem. The proposed solution is based on the 3D CNN, with a novel architecture featuring a Generative Adversarial Network used during training to improve network performance for unseen data.

All the proposed networks demonstrated a state-of-the-art performance compared to other corresponding methods reported in the literature on a number of assessment metrics. In particular, the proposed architecture for 3D motion field estimation has shown to outperform the previously reported manual expert-guided registration methodology.

# List of Abbreviation

| | |
|---|---|
| Adam | Adaptive Moment Estimation |
| DIR | Double Inversion Recovery |
| DM | Deep Match |
| CNN | Convolutional Neural Network |
| CRF | Conditional Random Field |
| CT | Computerized Tomography |
| DL | Deep Learning |
| EPE | End Point Error |
| FCCN | Fully Convolutional Neural Network |
| FCNN | Fully Connected Neural Network |
| GAN | Generative Adversarial Neural Network |
| GC | Generalized Charbonnier |
| KL Divergence | Kullback-Leibler Divergence |
| KLT | Kanade-Lucas-Tomasi Tracking |
| LBP | Local Binary Patterns |
| LIDAR | Light Detection and Ranging |
| LReLU | Leaky Rectified Linear Activation Function |
| LSGAN | Least Square Generative Adversarial Neural Network |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| MR | Magnetic Resonance |
| MRF | Markov Random Field |
| NLP | Natural Language Processing |
| PCA | Principle Component Analysis |
| RANSAC | Random Sample Consensus |
| RBM | Restrictive Boltzmann Machine |
| ReLU | Rectified Linear Activation Function |
| RNN | Recurrent Neural Network |
| SIFT | Scale Invariant Feature Transform |
| SGD | Stochastic Gradient Descent |
| SGM | Semi-Global Matching |
| SSD | Sum of Squared Difference |
| STN | Spatial Transformer Network |
| TDNN | Time Delay Neural Network |
| TPS | Thin Plate Spline |
| TRE | Target Registration Error |
| TRUS | Intra-Procedural Transrectal Ultrasound |
| VGG | Visual Geometry Group |

# List of Abbreviation

| | |
|---|---|
| $\beta$ | Constant |
| $\lambda$ | Scaler Constant |
| $\gamma$ | Smoothness Constant |
| $\boldsymbol{\vartheta}$ | Transformation parameters |
| $\epsilon$ | A small value in the range of $10^{-6}$ |
| $\boldsymbol{\theta}$ | The set of the latent variables in the Multilayer NN or CNN |
| $\nabla$ | Gradient Operator |
| $\Omega$ | Data domain |
| $b$ | Stereo Camera baseline |
| $d()$ | Disparity function |
| $d^b$ | Backward disparity |
| $d^f$ | Forward Disparity |
| $Disc(x;\theta^{(d)})$ | The discriminator notation in the GAN |
| $g$ | Charbonnier loss function |
| $\boldsymbol{E}$ | Camera essential matrix |
| $E(,;\theta)$ | Error energy function giving the latent variable set. |
| $E_D$ | Charbonnier loss function |
| $\mathbb{E}_{x\sim p_{data}}$ | The expectation when $x$ is sampled from the real data distribution in GAN |
| $\mathbb{E}_{x\sim p_{model}}$ | The expectation when $x$ is sampled from the fake data distribution in GAN |
| $f_L$ | Camera focal Length |
| $\boldsymbol{F}$ | Fundamental Matrix |
| $\boldsymbol{h}$ | Hidden feature or state. |
| $I$ | Image Intensity |
| $I_l$ | Left image |
| $I_r$ | Right image |
| $\boldsymbol{K}$ | Intrinsic camera parameters matrix |
| $l$ | $z$ motion component |
| $\ell_{GC}$ | Charbonnier loss function |
| lr | Learning Rate |
| $\ell_S$ | Smoothness loss function |
| $m^b$ | Backward mask function |
| $m^f$ | Forward mask function |

| | |
|---|---|
| $\boldsymbol{M}$ | 3x4 Homogenous projective matrix |
| $\boldsymbol{P}$ | Point in the 3D space |
| $Q$ | A matrix used in 3D reconstruction of 2D scene using disparity. |
| $\boldsymbol{Q}$ | 3D Displacement Field |
| $\mathbb{R}$ | The set of the real numbers |
| $\boldsymbol{R}$ | Angular orientation square matrix |
| $R$ | Radius of sampling in LBR |
| $\widehat{\boldsymbol{T}}$ | Motion Field |
| $\widetilde{\boldsymbol{T}}$ | Estimated Motion Field |
| $\boldsymbol{T_{proj}}$ | 3x4 Projection matrix |
| $Z$ | Object depth |
| $u$ | $x$ motion component |
| $\boldsymbol{U}$ | Input-to-hidden latent weight matrix (LSTM) |
| $v$ | $y$ motion component |
| $\boldsymbol{V}$ | Hidden-to-output latent weight matrix (LSTM) |
| $\boldsymbol{W}^{(L)}$ | Matrix represents the CNN layer's weights |
| $\boldsymbol{\omega}$ | Angular frequency |

# Table of contents

## List of Figures

## List of Tables

# Chapter 1 - INTRODUCTION

## 1.1 Motivation

Computer vision is a multidisciplinary area of science and technology, which is concerned with the processing and interpretation of visual information provided in the form of images and videos of various modalities and dimensionalities. It is tasked with the development of fundamental underlying techniques and algorithms as well as engineering solutions enabling the implementation of the computer vision methods in practical applications. It is frequently described as a science and technology that enables machines to visually perceive the world and interact with it, in some sense trying to mimic human visual perception. Some of the main computer vision areas include image and video restoration, 3D pose estimation, 3D scene reconstruction, modelling, data co-registration and augmentation, visual information retrieval and indexing, surveying and robot navigation, and object recognition and tracking, event detection, and finally, motion estimation. Many of the computer vision techniques developed up to now have found their way into practical implementations. More recently, computer vision provides key enabling methodologies for developing novel breakthrough technologies, including self-driving cars, easily accessible everyday medical diagnostics, commerce, and remote sensing, e.g., for inexpensive agricultural crop and livestock monitoring.

One of the most challenging problems in Computer Vision is the analysis of motion information embedded in image sequences. It can reflect the dynamics of the observed scene and cameras used for acquisition. Some examples of the information that can be retrieved from image sequences include optical flow, dense or sparse displacement field, disparity, camera ego-motion, scene geometry, and object detection and categorization. This information is essential for many practical applications, such as structure from motion (SfM), simultaneous localization and mapping (SLAM), 3D reconstructions, image registration, autonomous navigation, and many others. This research field is still one of the most challenging and active research areas in computer vision due to the wide range of possible critical applications and the fundamental algorithmic and implementation difficulties. The complexity of these algorithms typically increases with the increase of dimensionality of the problem, e.g., in dense high-resolution motion estimation for high-multidimensional sequences.

Motion applications span diverse fields and areas, for instance, the film and video industry (Seshadrinathan et al., 2010; Yue-Hei et al., 2014), medicine (De Craene et al., 2012 ), surveillance

(Ouyang and Wang, 2013), autonomous vehicle navigation (Menze and Geiger, 2015), biological science (Jain et al., 2014) are just a few examples to mention. Motion applications revolve around three main fields of study, video categorization, motion detection, and motion estimation. Since the formation of the computer vision in 1963 (Roberts, 1963) and despite the extraordinary progress in that field, the technical difficulties in matching the performance of the biological vision system (Hartley & Zisserman, 2003) were obvious. In the past, computer vision techniques generally used handcrafted features. This approach couldn't address the complexity and appearance variations of data for practical problems tackled. With the advances in the theoretical and experimental understanding of biological vision systems, the fundamental shift in computer vision has also been achieved (Medathati, et al., 2016). This progress is mainly due to Machine Learning (ML) and, in particular, Deep Learning (DL). This new approach allows the modern computer vision to achieve results comparable and, in some cases, better than a human observer.

ML key concept is to allow the model to learn from data (Marsland, 2014). Generally, ML aims to discover patterns present in the data to develop models and tools for the analysis and prediction. ML can produce a good representation of visual data (Medathati, et al., 2016).

One of the recent most successful machine learning approaches is Deep learning (DL) (LeCun, et al., 2015). This type of learning's main concept is based on the fact that an algorithm learns an effective representation of the data autonomously and directly from the available training data without any prior, possibly biased assumption about optimal representation imposed by a human expert. This approach is significantly different from previous ones, where an expert decides what type of data representation or feature set (so-called handcrafted features) should be used to perform a specific ML task. The hierarchical representation that provides a logical internal representation of images (LeCun, et al., 2010) was proposed, with pixels assembled into edges and edges into patterns and patterns into parts, and part into objects and objects into scenes. These hierarchical representations suggest a multistage vision architecture approach.

Nowadays, DL became one of the most active research areas since the term first coined in the eighties (Dechter, 1986). Moreover, DL made significant progress in big data feature learning like ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Krizhevsky, et al., 2012) and many other learning tasks (Girshick, 2015; Dosovitskiy, et al., 2015; Karpathy, et al., 2014; LeCun, et al., 2010). The DL model depth is the key aspect of the strategy in this class of machine learning model. The depth in this context represents the longest path between the input and the output of the model. Controlling the depth of the model offers two benefits; first, it addresses the hierarchical abstraction of representations and, second, feature reuse due to multipath the model allows (Bengio, et al., 2013).

Convolutional Neural Networks (CNN) represents a broad class of DL models that are flexible enough to fit this multistage architecture (LeCun, et al., 2010). Each stage of this model uses trainable arrays called feature maps. It could be a 1D array of features as in the case of the audio signal, 2D as in the case of images, or even 3D as in the case of volumetric data, e.g., CT scans.

The networks proposed in this work addresses the practical implementation of a real situation where real-time inferencing is required, including motion detection and dense motion field estimation. These tasks influence many computer vision applications, like human-machine interfacing, segmentation, anomaly detection, autonomous vehicle applications, augmented reality, and 3D reconstruction. In this work, training methods are changing between supervised and unsupervised learning based on the type of problem in hand and the datasets used.

Based on the significant reported advances in machine learning, it is expected that it should be possible to better leverage these techniques for computer vision motion detection, recognition, and estimation problems. It was postulated that proposed novel algorithms should be based on a machine learning methodology as opposed to more classical approaches based on the analytically constructed models of motion, with motion models derived directly from data. It was also decided that the primary toolset used on the project should be deep learning as it provides a proven and convenient approach for information extraction directly from the data without the need for prior data models.

## 1.2 Motion Detection and Estimation Challenges

Generally, motion detection/estimation experiences technical difficulties when performed. These difficulties are due to the fact that the motion as a phenomenon needs to be extracted directly from the image sequence. One of the well-known technical problems that need to be addressed for motion estimation is the concept of the apparatus problem discussed in Section 2.6. This problem introduces ambiguity in the analysis. Compared to other computer vision problems, like object recognition (classification) or semantic segmentation, motion detection/estimation faces other technical difficulties in its modeling. For instance, motion events may rarely appear in the image sequence, which causes the data-based model optimization to be an expensive task.

Instantaneous estimation of the motion in a scene is still an intractable problem due to the inherent representation of the image. In addition, modeling the object motion itself should imply some sense of simultaneous classification and segmentation on a pixel level. This awareness of the object and the scene requires another more critical operation of the scene's local and global representation understanding.

Previously, motion detection and estimation were performed on perfect quality images with even illumination across the images that were used for proof of concept (Scharstein, et al., 2014). However new realistic dataset has increased the challenging bar. For example, sun reflection on a smooth surface and transparant surfaces increase the challenge of computer vision research in this field of study (Geiger, et al., 2013).

## 1.3 Aim

The overall research aim of this work has been to develop novel convolutional neural network models that provide state-of-the-art solutions for a selection of different motion analysis problems in computer vision. More specifically, the work aims at developing novel deep learning methods that can improve applications related to motion in computer vision, applications like (i) motion detection, (ii) dense 2D motion field estimation, and (iii) dense 3D motion field estimation. A set of corresponding practical computer vision problems that are related to these applications have also been selected. This has been done to provide practical solutions, often necessitating the resolution of challenging application-specific problems. Suitable platforms will also support this approach for validations against already reported methods on standard datasets. The corresponding problems that will be considered in this project including: (i) eyeblink detection for motion detection, (ii) stereovision disparity estimation for dense 2D motion field estimation, and (iii) deformable registration of volumetric CT data for dense 3D motion field estimation.

## 1.4 Objectives

Each of the three problems described in the previous section defines a set of unique objectives that need to be addressed to devise a successful solution for the posed aim.

The main objectives, which need to be considered to successfully solve the **eyeblink detection,** are as follows:

- The eyeblink detection solution should cope with distortions present in real data, including varying illumination conditions, changing the video frame rate, and the head pose. It should also deal with motion blur and unknown eye position.
- While various reported in literature methods often use annotated data, with eye positions are known, these methods' performance may deteriorate when using separate eye detectors to

identify eye positions when using unannotated data. Therefore, unannotated data with unknown eye positions to train the proposed deep learning models should be used.

- The input data should be arranged to simulate a typical output from a face/eye detector pipeline, with each eye image cropped differently from frame to frame (including consecutive frames). For such cases, motion detection algorithms based on classical optical flow estimation would not work.

For the main objectives, which need to be considered for **disparity estimation,** are as follows:

- Since disparity is a regression problem, the proposed convolutional neural networks should also solve a regression problem.
- In stereovision, consistency of the disparity computed between images is one of the main requirements; therefore, the proposed solution should generate simultaneously forward and backward disparities.
- The ground truth data is difficult or, in some cases, impossible to obtain, and therefore unsupervised learning framework should be implemented.
- The occlusion is a typical problem in stereovision, making the situation challenging, affecting estimation accuracy and computation time. Therefore, the proposed methods should be able to explicitly and efficiently model the occlusion maps for each image.
- The loss function should address all the above problems as well as impose the disparity smoothness constraints, excluding parts of the disparity maps corresponding to depth discontinues in the scene.

In the case of **3D CT registration**, the main objectives are,

- As in the case of the 2D disparity, the proposed deep network should solve a regression problem as well as be trained using an unsupervised methodology.
- The 3D registration imposes significant memory demand. Subdividing volumes into smaller sub-volumes typically reduces accuracy due to correspondence problems. The proposed network should be able to accommodate the entire volume enabling 3D motion field estimation in one pass.
- The key problem for developing 3D CT registration deep networks is the limited availability of the training data. Therefore, it is postulated to incorporate, within a proposed deep architecture, the Generative Adversarial Neural, addressing this limitation.

## 1.5 Novel contributions

This thesis presents a number of novel approaches for selected computer vision motion analysis problems. The main focus of the work has been on research and development of original deep learning architectures addressing these important and challenging problems. The novel solutions have been proposed for all three case studies considered in the thesis. In all cases it has been shown that the proposed novel approaches improve performance when compared to the existing state-of-the-art methods.

For the eyeblink detection, two novel architectures have been developed. In the first approach, spatial and temporal networks are designed independently, with the processing pipeline resembling an image captioning architecture but adopted for the motion detection problem. The second novel approach uses a network model, which is based on the Time-Distributed Convolutional Neural Networks with added Long-Short Term Memory layers. The key advantage of the proposed solution is its resilience with respect to the errors introduced at the eye detection stage performed for each image frame. The performance of the developed deep learning models showed a significant improvement over traditional eyeblink detection approaches used for this problem.

For the stereovision disparity estimation, the key novelties are in the proposed direct modelling of occlusion maps, concurrent estimation of the forward and backward disparities with the imposed robust smoothness constraint, and an unsupervised training regime. The model trained with the occlusion aware loss function has shown better error characteristics than a model trained with agnostic loss function. The qualitative tests of the developed model have showed the capability of the model to obtain disparity maps of stereo images with different camera specifications. This indicates that the model can be possibly generalized to other data not used in the experiments.

For the 3-dimensional CT registration, one of the main contributions is proposed novel network architecture. It allows the network to operate efficiently on a large 3D data set. Another novelty is proposed use of an embedded GAN, during training of the main network, to provide further regularisation of the regression model. Quantitative experiments showed improved performance of the registration network when GAN was using during training compared to the network trained without GAN. However, in either cases, the proposed approach outperforms the previously reported methods, including manual registration performed by experts.

## 1.6 Thesis Structure

The thesis is arranged in a logical order with the increasing complexity of the investigated applications. An introduction to computer vision focused on motion detection and estimation is presented first.

Subsequently, a review of deep learning concepts, ground rules, and algorithms, with a focus on the convolutional neural network, is provided. After explaining the main computer vision and deep learning concepts, the three selected applications' work is reported in three successive chapters. In each of these chapters, the specific problem is first explained, and the existing solutions are reported. Next, proposed novel solutions are described in detail, and the quantitative evaluation results are reported, with an emphasis on critical comparison with previously proposed methods. The last chapter provides an overall summary of the work reported in the thesis.

Chapter 2 provides an overview of the motion-related topics in computer vision. The chapter starts with the camera models to pave the way to the motion estimation methodologies. The two main motion estimation methodologies, parametric, and non-parametric, are reviewed. The concept of motion detection, 2D, and 3D motion field estimation problems are explained.

Chapter 3 provides an introduction and review of Deep Learning concepts. It starts with a description of the essential building blocks of deep learning algorithms. Then, different regularization techniques and training concepts are explained. Subsequently, popular deep classification network architectures are reviewed with emphasis on their main conceptual innovations. Furthermore, networks explicitly designed to work with sequential data are examined. The chapter ends with a review of the Spatial Transformation Network and Generative Adversarial Network.

Chapter 4 describes research on eyeblink detection, selected as an example of a motion detection problem. Two different approaches are proposed. The first method uses two networks, spatial and temporal, which are trained separately. The second model uses a single network trained in an end-to-end fashion. In both cases, the main objective was to develop models that can run online in real environments. The reported results show that the proposed new network models outperform methods previously reported in the literature.

Chapter 5 describes the stereovision disparity estimation work, selected as an example of the 2D dense motion field estimation problem. The proposed novel approach is designed to explicitly model occlusion in both stereo images. The model performs forward and backward disparity estimations in two parallel channels, improving the estimated disparity map's overall consistency. The proposed method is computationally efficient and provides competitive accuracy results when compared to other reported methods.

Chapter 6 describes research on 3D volumetric CT registration, selected as an example of the 3D dense motion field estimation problem. The proposed method uses a novel deep network architecture, addressing the data's high dimensionality, and incorporates a Generative Adversarial Network to

further improve network regularization. The reported results of quantitative evaluation and comparison with other methods show state-of-the-art performance.

Chapter 7 provides an overall summary of the reported work, with a list of novel contributions, research conclusions, and plans for future work. It reviews the main results and points out the state-of-the-art achieved in this work.

# Chapter 2 - MOTION ANALYSIS

## 2.1 Introduction

This chapter paves the way to the conceptual and mathematical background regarding the motion in images. It first describes camera models that represent the sensor of the 3D world scene. The camera transforms the 3D geometry of the scene into a 2D representation. Two motion models that address the rigid and non-rigid motion are presented. While the first model is based on geometrical relations to estimate the motion, the later model is based on pixel correspondence estimation. Then, the definition of motion detection, motion, 2D, and 3D geometry are described. In the formulation of this chapter, Euler representation has been adopted, and no quaternion representation has been utilized.

## 2.2 Motion from Images Challenges

In many applications, obtaining motion information from images was and still is one of the main topics in computer vision. Motion from the image is still a topic that is hard to solve. Generally, to obtain motion information, the required image's specification must be set first to adapt the algorithm to a tight range of the expected motion information retrieved. For instance, an algorithm that can calculate the terrestrial depth from stereo images will not fit to calculate the depth of the scene for driving a car. This problem can be applied to many other motion-related phenomena from images. Focussing on the blink as a motion detection problem, it was required to build a particular dataset to address the problem by training a model after modelling the case as a sparse function of time. For that, a complete dataset has been built with ground truth. In the case of disparity, optical flow, and medical image registration related to a real scene, there is no ground truth available. Lacking the ground truth has led the study to depend on unsupervised learning to achieve the required task. Unsupervised learning is one of the most complex and challenging solutions in the DL field.

To appreciate the motion extraction task from image, this chapter is addressing the concepts and theoretical backgrounds of the camera that inherent the idea behind the hidden motion information within the image and discuss how this information can be useful to obtain other representation that is buried within the spatial representation of the 3D scene. Couple of the challenges that related to the motion from image can be listed below as:

### 2.2.1 Motion Detection

Motion detection is to identify motion represented by changes in the intensity of a region between two or more consecutive video frames or time instances. The spatial changes in the intensity can be on different levels. For example, the detection may be performed per pixel, edge, or region intensity changes depending on the type of the problem at hand. The problem can be as simple as subtracting two images where the difference represents a disturbance in the intensity level related to motion.
Although it may seem a simple problem, a small perturbation in the intensity that may be induced from the object's small movement or even the inherent noise in the camera may produce the wrong detection. Furthermore, changes in the image intensity between two consecutive frames, blur, unregistered images, light reflection, transparency, and other situations could fail any algorithm that is usually designed to address the perfect condition.
Thus, the motion detection from image sequence has difficulties that need to be addressed for successful model implementation.

Motion detection is used to automate the video surveillance system recording. Techniques like background subtraction may not be sufficient though in uncontrollable lighting conditions. Several noise reduction and image enhancement techniques like histogram equalization may be employed before the motion detection algorithm can be applied. The motion detection is implemented utilizing three main methods; background subtraction, temporal differencing, and optical flow (Huang, 2010; Huang, et al., 2018; Parmar & Chitaliya, 2013). The other applications of motion detection include traffic monitoring (Hao, et al., 2012), segmentation (Wei, et al., 2011), and anomalous detection (Roberts, et al., 2009).

### 2.2.2 Motion Classifications

In motion classification, the task is to categorize the sequence of frames extracted from video to its correct video class (Fernando & Gould, 2016). The video class could be ‘running class’ or ‘jumping class,’ and the video related contains materials related to that activity. The difference between image classification and video classification is that image classification focuses on the subject in the image if the image contains a cat or dog with a minimum focusing on the context. While in video classification, the task is more challenging because a model needs to exploit any information related to the activity. For instance, to classify football footage that the game is on run or it been ended or stopping for some reason, the video classification model should analyse the context related to the video status. Therefore, using a limited sequence of video frames sampled at a fixed sampling period from videos can be utilized for model training (Karpathy, et al., 2014).

## 2.3 Motion Analysis Methods

Images are essential sources of information about a static scene structure, whereas image sequences or videos contain much richer visual cues that can be useful for representation, modeling, or recovery of the scene dynamics. Humans have a remarkable ability to detect subtle motion, with the human visual system being sensitive to small but rapid changes than to more significant but slower image variations. However, motion modeling, estimation, detection, and analysis tasks remain a considerable challenge in computer vision that requires advanced image processing techniques and significant computational resources. Recently, there has been a renewed significant interest in quantitative motion estimation techniques, which could be accredited to the rapidly growing computing hardware capabilities regarding both the storage and processing speed.

For sequences of 2D images, the optical flow is frequently used for motion representation. It is used as a proxy for the representation of the corresponding 3D motion in the scene. Due to the 2D image formation process's inherent properties, there is a persistent difficulty revealing a real 3D motion based on the estimated, from an image sequence, 2D optical flow. The apparent motion is usually related to the object's optical flow motion to reflect the fact that this projection is not the complete motion.

It will be shown that sometimes the real 3D motion can be retrieved in very special conditions like rigid motion. For instance, a rigid body's complete motion can be described geometrically by translation and rotation matrices. However, it may not work in all cases, as in the rotating sphere where geometrical representation is not effective. Moreover, retrieving a dense representation of motion is generally quite difficult.

Motion analysis is fundamental for many applications in computer vision and machine learning. First, it provides a strong relationship between image objects in spatiotemporal cues. These cues can be utilized in applications like object detection. The motion can also be detected if the image pixel moved through the frames, as in eye blink detection or the surveillance system. Furthermore, it can be classified when a group of pixels achieved specific transition within a group of frames or estimated when objects achieve coherent displacement in an image sequence as in optical flow and disparity estimation. Motion in an image can also be utilized in 3D reconstruction and segmentation.

## 2.4 Camera Model

The camera model describes the projection of the 3D objects from a 3D scene onto the camera's 2D image plane. Figure 2.1 shows an interpretation of a simple pinhole camera model, with the camera and the world coordinate systems co-located.

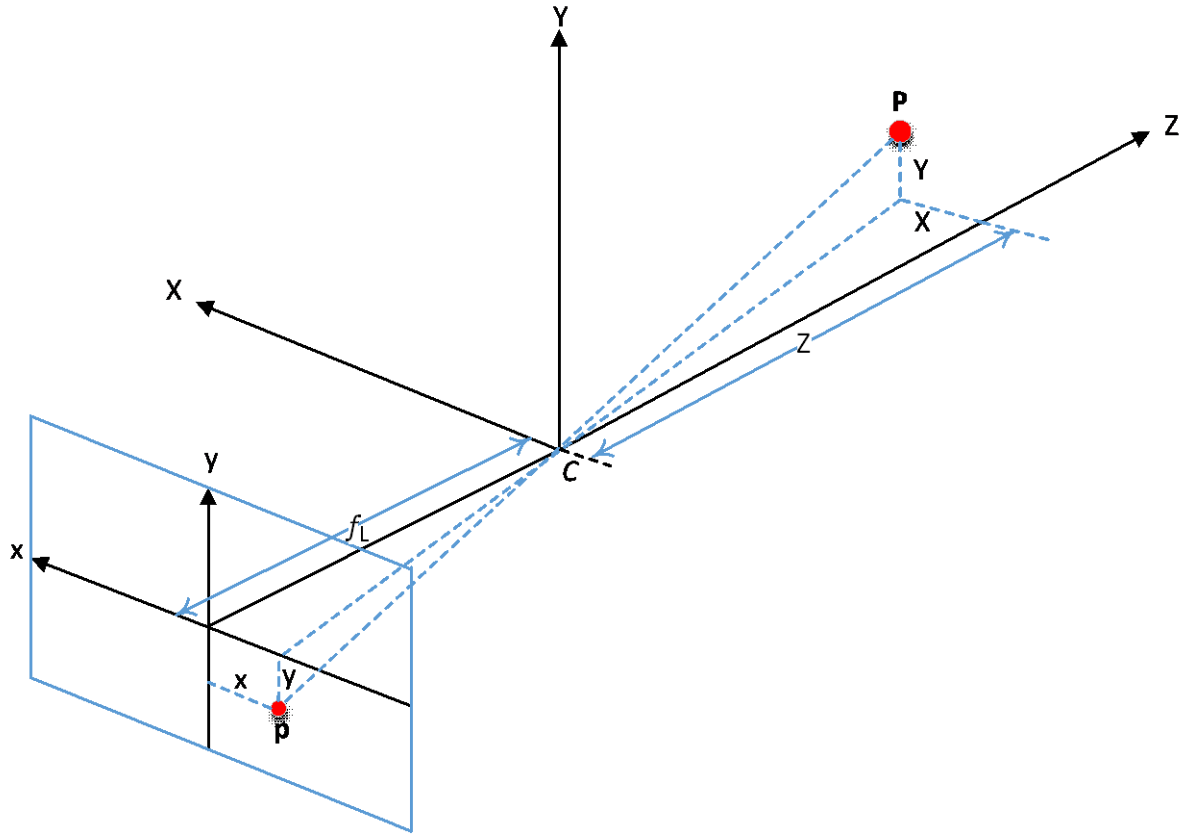



Figure 2.1: Graphical representation of the pinhole (or perspective projection) camera model (Wang, et al., 2002), C is the camera optical center (i.e., pinhole), f is the focal length.

Figure 2.1 shows a 3D point **P** is projected on the 2D point **p** located in the image plane (also known as Retina) with the optical rays passing through the pinhole (camera optical center *C*). The distance between the image plane and the camera center is the distance referred to as the focal length ($f_L$). Assuming that the image plane is parallel to the XY plane and the camera center is located at the 3D world coordinate system's origin. Then, using similar triangles, the correspondence between the 3D scene point $(X, Y, Z)$ and corresponding projected point ($p'$) located in the 2D image plane, can be obtained using the pinhole camera model shown below, Figure 2.2. Note that both Figure 2.1 and Figure 2.2 are geometrically compatible pinhole models. The difference is that the image plan in the second figure is in front of the optical center.

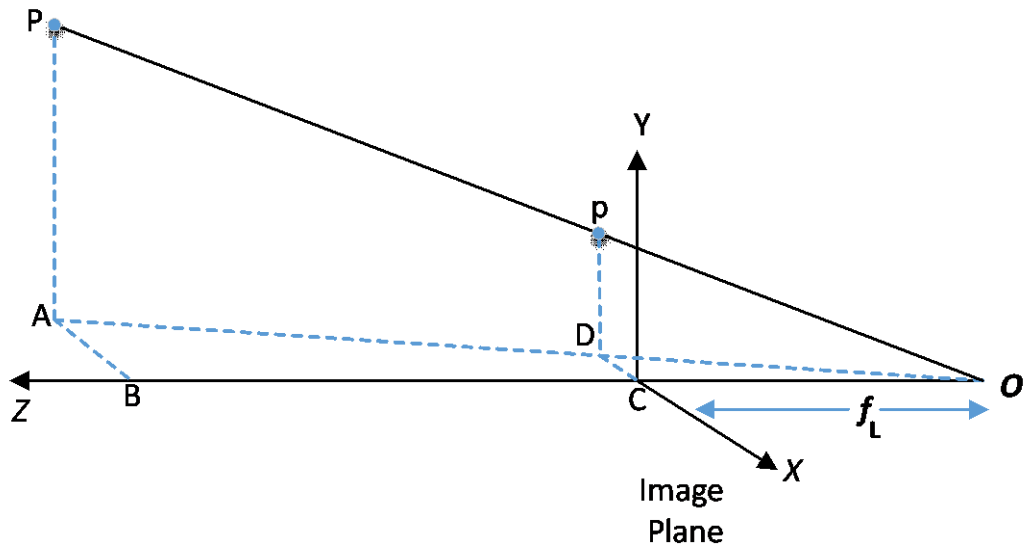



Figure 2.2: Pinhole model.

Here, $Z$ is perpendicular on the view plane or the image plane and crossing it on the image centre C as well as the camera focal point $O$. The displacement between the camera focal point $O$ and the image centre C is defined as the focal length $f_L$. The projection ($p'$) of the point $\boldsymbol{P}$ on the image plane can be obtained as follows:

From Figure 2.2 above:

$$|AB| = X, |PA| = Y, |BC| = Z, |DC| = x, |p'D| = y, |OC| = f_L$$

the triangles ABO and DCO are congruent, therefore,

$$\frac{|DC|}{|AB|} = \frac{|OC|}{|OB|} \tag{2.1}$$

which lead to

$$x' = \frac{X}{1 + Z/f_L} \tag{2.2}$$

Similarly,

$$\frac{|p'D|}{|PA|} = \frac{|OD|}{|OA|} = \frac{|OC|}{|OB|} \tag{2.3}$$

leads to

$$y' = \frac{Y}{1 + Z/f_L} \tag{2.4}$$

With $Z/f_L >> 1$, both of Equations (2.2) and (2.4) can be approximated to :

$$x' = \frac{f_L X}{Z} \quad and \quad y' = \frac{f_L Y}{Z} \tag{2.5}$$

This representation is known as the **perspective** projection, and it can be expressed in a homogeneous form as:

$$\lambda \boldsymbol{p}' = \begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} f_L & 0 & 0 & 0 \\ 0 & f_L & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

where $\lambda$ is scaler constant. The 3x4 matrix is also called the projection matrix after dividing by the focal length.

$$\boldsymbol{T_{proj}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f_L & 0 \end{bmatrix} \tag{2.7}$$

Equation (2.6) can be writing in more reviling form as:

$$\lambda \boldsymbol{p}' = \begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} f_L & 0 & 0 \\ 0 & f_L & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.8}$$

Or Equation (2.8) can be expressed in compact form as:

$$\boldsymbol{p}' = \boldsymbol{K}[\boldsymbol{I}\ \boldsymbol{0}]\boldsymbol{P} \tag{2.9}$$

where the 3x3 matrix represents the intrinsic camera parameters, and the 3x4 describes the extrinsic camera parameters. The first three columns of the extrinsic matrix are identical to the identity matrix, which is when both the camera coordinate system and the world coordinate system coincide.

The generalised form of Equation (2.9) can be obtained by including other intrinsic and extrinsic parameters of the camera, as reported in (Zhang, 2012):

$$\lambda \boldsymbol{p}' = \begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} f_L/s_x & s & c_x \\ 0 & f_L/s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.10}$$

This can also be represented in a compressed form as:

$$\lambda \boldsymbol{p}' = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{T}]\boldsymbol{P} = \boldsymbol{M}\boldsymbol{P} \tag{2.11}$$

where $\boldsymbol{M}$ is the camera projection matrix that models the geometric image formation. The 3x3 matrix $\boldsymbol{K}$ describes the intrinsic camera parameters matrix, which models misalignment of the image plane coordinate system and the image coordinate centre and the discretisation of the image coordinates. More specifically, $1/s_x$ and $1/s_y$ are pixels aspect ratio in the x and y direction of the image plane (Tekalp, 2015). s is the skewness parameter due to non-orthogonal camera axes. Translation $c_x$ and $c_y$ are the image centre coordinates with respect to the camera centre (Jain, et al., 2003). The other terms of Equation (2.11) include $\boldsymbol{R}$ and $\boldsymbol{T}$, which will be discussed later, related to the rotation and translation, respectively, which model the camera pose and its coordinates with respect to the scene coordinates.

In Euler representation**,** $\boldsymbol{R}$ is an angular orientation square matrix that can be specified by three angles:1. rotation $\omega$ about the x-axis or the pitch (vertical angle) of the optical axis

2. rotation $\varphi$ about the y-axis or the yaw (horizontal angle) of the optical axis.

3. rotation κ about the z-axis or the roll or twist about the optical axis.

R can be expressed as (Jain, et al., 2003):

$$R = \begin{bmatrix} \cos\varphi\cos\kappa & \sin w\sin\varphi\cos\kappa + \cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa + \cos\omega\sin\kappa & \cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{bmatrix} \tag{2.12}$$

It can be seen from Equation (2.7) that as the focal length $f_L$ increases, that is as the focal point O moves along the principal axis (Z-axis), an **orthographic** projection will be obtained. The assumption allows as a special case of the perspective projection. Hence Equation (2.7) can be written as:

$$\boldsymbol{T}_{\boldsymbol{proj}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

Having the third column in Equation (2.12) shows that the depth information $Z$ are usually lost during the spatial projection, which is when the object is far from the camera, and the lens is considered to be thin. In this situation, there are no significant depth variations. The perspective projection can be

approximated by orthographic projection or what is also referred to as parallel projection, and Equation (2.12) yields:

$$\boldsymbol{x'} = \boldsymbol{X}, \quad \boldsymbol{y'} = \boldsymbol{Y} \tag{2.14}$$

Or in matrix form as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.15}$$

In Equation (2.7) above, all the coordinates are in cartesian, and it is valid if the variation in the depth of the object within the image can be neglected. Note that both the perspective and the orthographic projections are many-to-one mapping since only the first object point seen in the ray or the line of sight is represented by one point on the image plane.

The **weak-perspective** projection or the **scaled-orthographic** projection is a middle case between the two, the perspective and the orthographic projection. It represents a compromise between the realism of the prospective projection and the simplicity of the orthographic projection, and it can be defined as (Tekalp, 2015):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{f_L}{Z} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.16}$$

## 2.5 Camera Setup and Motion Models

In this section, two motion models are presented, namely, the projective and the apparent models. In the projective model, the relative motion between a camera and a scene is defined by 3D translation and rotation along with the camera's intrinsic parameters needed to estimate a real representation of the motion. The projection of the 3D motion scene on the 2D image plane is a geometrical transformation by which it could be related to the 3D motion of the camera itself, calibration changes of the camera parameters, independent motions of different objects within the scene, or a combination of all these factors.

For the second model, the apparent motion model, the specification of the image acquisition system is not required. This motion estimation method does not provide retrieval of the 3D motion, and it is mainly related to the motion in 2D in the image plane generated from the captured 3D scene motion, hence the term apparent motion. The authors in (Yao Wang, 2001) address situations where the projective model is not applicable, and it is more likely that time-varying image intensities can be processed in the apparent motion model. Apparent motion model examples are dense optical flow and points correspondence that will be explored in the following subsections.

This section considers a few projective motion models, which include moving the camera in a static scene, moving scene with a static camera, and motion scene with a moving camera. These cases are similar both in the problem setting and the formulas describing relevant solutions.

### 2.5.1 Moving Camera in Static Scene

The geometry of this model is shown in Figure 2.3. Here, camera1 is referred to as the camera in one position at time $t1$ and the camera2 is referred to as the camera in another position at time $t2$. The configuration is identical to a stereo vision setup, and the motion estimation corresponds to pose estimation. In stereo vision with two cameras, the image pair can be taken at the same time, the static scene requirements can be lifted, and the objects can move within the scene accordingly (Tekalp, 2015).

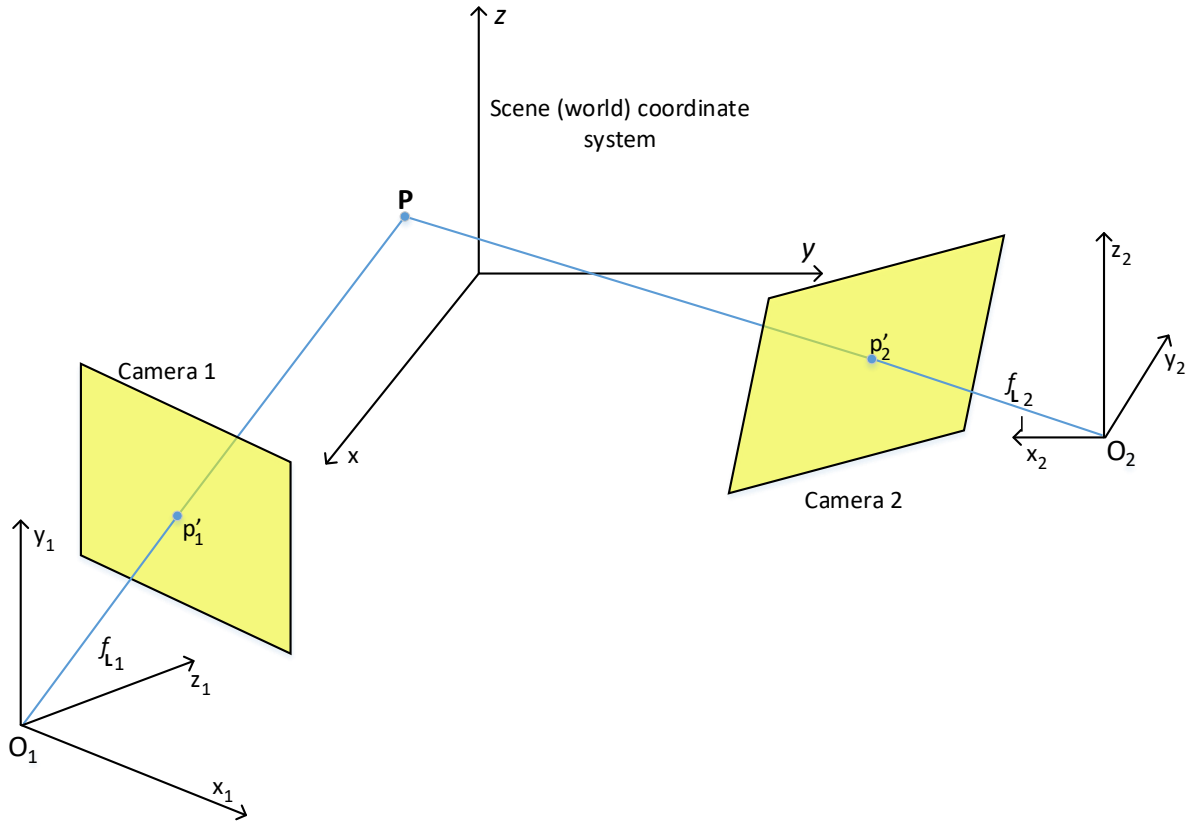



Figure 2.3: Camera motion with a static scene or stereo vison (Change shape pinhole to be behind the image plane) (Tekalp, 2015).

The projected image-plane coordinates of the feature points can be expressed in homogenous coordinates Equation (2.5), $\mathbf{X}$ is the feature point in the scene (world) coordinates, $\mathbf{M}$ is 3x4 homogenous projective matrix for the second camera, and $\lambda$ is scaler. The relative pose between camera 1 and camera 2 can be modeled by the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{T}$.

$$\boldsymbol{p}' = \mathbf{K}[\mathbf{R}\ \mathbf{T}]\mathbf{P} \tag{2.17}$$

Thus, the term $[\mathbf{R}\ \mathbf{T}]$ provide the necessary pose (motion) estimation in 3D.

### 2.5.2 Static Camera with Rigid Scene Motion

If a 3D feature point $\boldsymbol{P}$ located in a rigid body moves from time $t$ to a new point position $\boldsymbol{P}$' at time $t$', the projection of the line connecting $\boldsymbol{P} - \boldsymbol{P}$' into the image plane can be denoted by the points $p$, $p$', respectively, Figure 2.4.

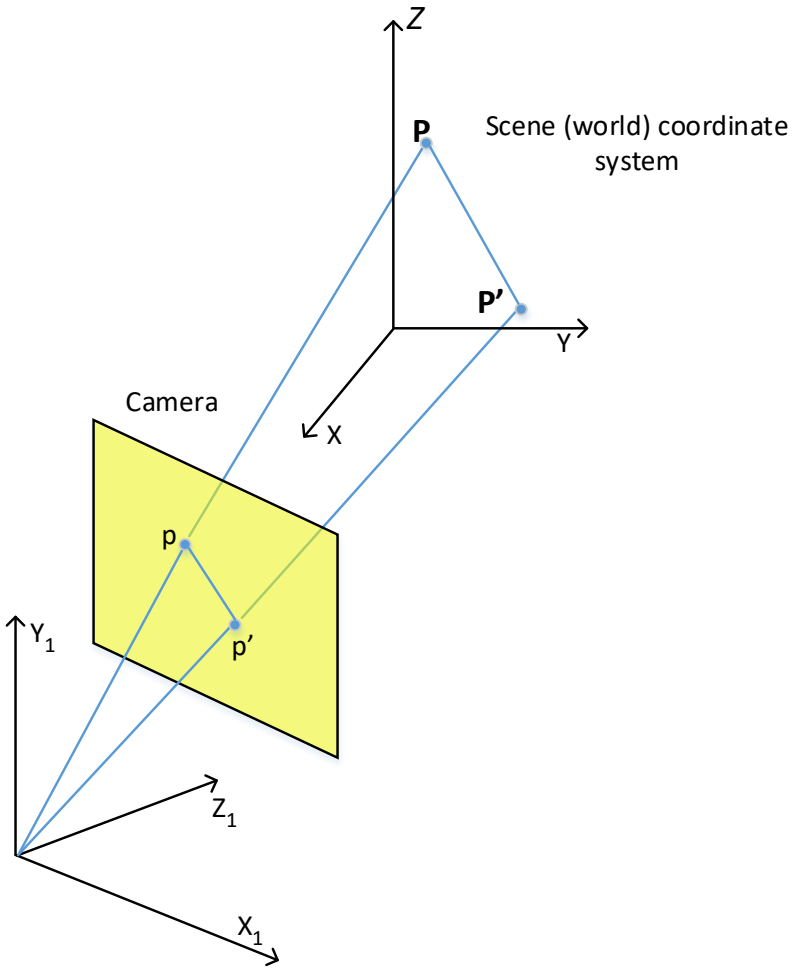



Figure 2.4: Projection of the 3D scene motion vector onto the image plane.

Considering the motion of each object point or pixel $\boldsymbol{P}$ independently, if the entire field of view contains an object, or of the object segmented, the relative 3D positions of all object points at time $t$ and $t'$ are related by a single rotation matrix **R** and translation matrix **T** as:

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{RP} + \mathbf{T} = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{21} & r_{22} & r_{32} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{2.18}$$

where **P** and **P**' are the coordinates of an object point at time $t$ and $t$' , respectively. (Yao Wang, 2001).

The homogeneous representation of Equation (2.17) can be expressed as:

$$p = \begin{bmatrix} r_{11} & r_{11} & r_{11} & T_1 \\ r_{11} & r_{11} & r_{11} & T_2 \\ r_{11} & r_{11} & r_{11} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{P} \tag{2.19}$$

The setup is similar to the case of the two-camera scenario. To obtain the projection of $p$ the point needs first to be transformed using Equation (2.10), then apply Equation (2.8).

## 2.6 Projected 3D Rigid-Motion Models

In kinematics, two types of 3D motions can be classified as rigid and non-rigid. In the rigid motion, the relative distance between points in the 3D scene persevered as the scene evolves in time. Below are common rigid 3D projections related to Section (2.2) described previously and their related transforms.

### 2.6.1 General Model (Depth Map)

If no assumption about the scene structure is considered, the depth of the object represented by the third coordinate $Z$ can be considered as an independent variable, then projecting the 3D point from the scene on the image plane at the point $p'$ at $(x', y')$ yields (Jain, et al., 1995):

$$x' = \frac{f_L X'}{Z'} = \frac{f_L(r_{11}X + r_{12}Y + r_{13}Z + T_1)}{r_{31}X + r_{32}Y + r_{33}Z + T_3} \tag{2.20}$$

$$y' = \frac{f_L Y'}{Z'} = \frac{f_L(r_{21}X + r_{22}Y + r_{23}Z + T_2)}{r_{31}X + r_{32}Y + r_{33}Z + T_3} \tag{2.21}$$

Dividing both the numerator and denominator of both expressions by $Z$ yields:

$$x' = \frac{r_{11}x + r_{12}y + f_L r_{13} + f_L T_1/Z}{r_{31}x + r_{32}y + r_{33} + T_3/Z} \tag{2.22}$$

$$y' = \frac{r_{21}x + r_{22}y + fr_{23} + fT_2/Z}{r_{31}x + r_{32}y + r_{33} + T_3/Z} \tag{2.23}$$

Here the value of Z, which is the depth, or the object distance from the image plane, appears as a parameter in the estimation of the $x'$ and $y'$ values. The 2D image motion vector (displacement or optical flow) is constrained with the 3D motion parameters that include the three rotation and the three translation. The depth contributes to the exact estimation of the motion vector in 2D.

### 2.6.2 Homography (Perspective Model)

The object points free depth parameters can be reduced substantially by modeling a moving object structure by a planer or piecewise planar non-deformable surface. The model for this described case is a 3D feature set or points that constructing the plane:

$$aX + bY + cZ = 1 \tag{2.24}$$

where the vector $[a\ b\ c]^T$ is the normal vector of the plane. With the right-hand side of Equation (2.23) is unity; the 3D displacement model of Equation (2.10) can be rewritten as (Tekalp, 2015):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{T} = \mathbf{R}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{T}\left[\begin{bmatrix} a & b & c \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right] = \mathbf{H}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.25}$$

where $\mathbf{H} = \mathbf{R} + \mathbf{T}[a\ b\ c]$. Applying Equation (2.16) to project the 3D scene into the 2D image plane, the homography relations can be obtained:

$$x' = \frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + h_9} \tag{2.26}$$

$$y' = \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + h_9} \tag{2.27}$$

Equations (2.17-2.18) is known as the perspective model. If $\mathbf{T} = \mathbf{0}$, Equation (2.16) can be reduced to $\mathbf{H} = \mathbf{R}$ and provides the necessary mapping between two images (which includes the intrinsic

and extrinsic camera rotations) regardless of the geometry of the scene. Two frames can be related by this homography relation if and only if (Hartley & Zisserman, 2003):

1. the same 3D planar surface is viewed by these frames with different rotation and translation of the camera.
2. the frames captured from the same camera that can rotate around its optical centre.

## 2.7 Observational or Apparent Models

Observational or Apparent motion tackle the non-rigid motion of the scene. In an image sequence, the first assumption for motion estimation is the unchanged or small changes of the object properties (e.g., color or intensity) within a video, which can hold the difference from frame to frame. Thus, a high correlation between frames in the direction of motion allows estimation of the motion accordingly (Tenbrinck, et al., 2013). The second assumption suggests that the motion field is generally smooth and un-suddenly.

The observational model exploits how the eyes observe variation in both color and intensity to detect motion (Gibson & Bovik, 2000). Optical flow is the apparent motion of the brightness pattern. While the motion field assigns a 3D motion vector to each point in the space, optical flow shows its projection on the 2D image. Consider the point $P$ in Figure *2.5*, with intensity $I$ at time t and located at contour C with isometric intensity, iso = same, metric = measure, or contour with an equal value of intensity. The point $P'$ represents its correspondent point at the time $t + \delta t$ located at the contour C' with same intensity value as C. However, the correspondences between points located on C and C' are not clear, and it is not easy to answer whether they relate to the same point, especially when the two contours are not looking similar. The implication is that the optical flow cannot uniquely be determined by the local information around $P'$ and further assumption is needed to estimate it.

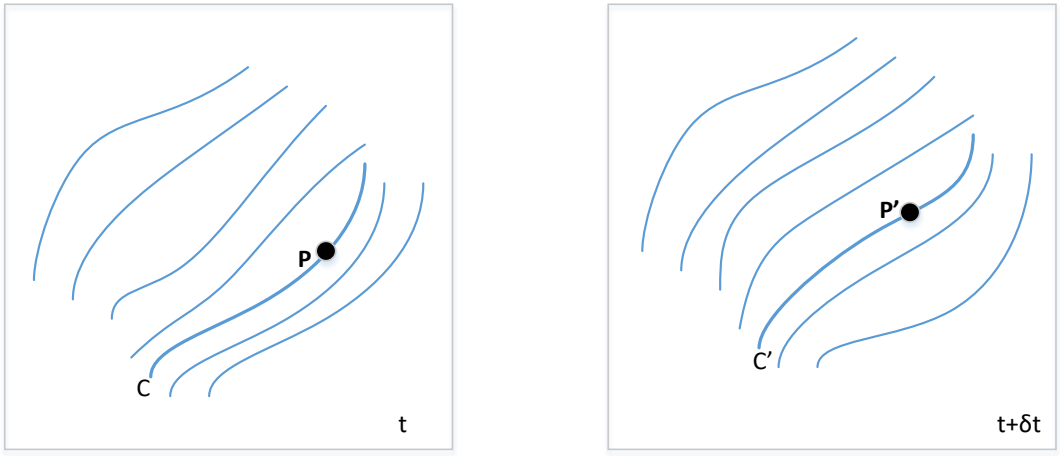



Figure 2.5: Optical Flow Observational Model. It is not easy to consider that point P' in the second image is related or represents the exact correspondence to P in the first image at t.

Now, assuming that the intensity at the points $P$, $P'$ are equal and $u(x, y)$ and $v(x, y)$ are the $x$ and $y$ component of the optical flow vector at that point, then the following constraint can hold for a small value of $\delta t$:

$$I(x, y, t) = I(x + u\delta t, y + v\delta t, t + \delta t) \tag{2.28}$$

The intensity usually varies smoothly with respect to $x$, $y$, and $t$. So, the right hand of Equation (2.28) can be expanded using the Taylor series as:

$$I(x, y, t) = I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} + e \tag{2.29}$$

where $e$ represents the second and the higher-order terms in $\delta x$, $\delta y$, and $\delta t$. Now, eliminate $I(x, y, t)$ from both sides, divide by $\delta t$ and taking the limit as $\delta t \longrightarrow 0$, the above equation can be written as:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = I_x u + I_y v + I_t = 0 \tag{2.30}$$

Or it can be expressed as:

$$\nabla \mathbf{I}^{\boldsymbol{T}} \boldsymbol{v} + I_t = 0 \tag{2.31}$$

where $\nabla \mathbf{I} = \left[\frac{\partial \boldsymbol{I}}{\partial \boldsymbol{X}} \quad \frac{\partial \boldsymbol{I}}{\partial \boldsymbol{y}}\right]^{\boldsymbol{T}}$ is the spatial gradient vector of $I(x, y, t)$ and $\boldsymbol{v} = [u \quad v]$ is the velocity or the flow vector. Equation (2.30) is called the optical flow constraint equation. The $u$ and $v$ of the optical flow vector are defined as $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$ . It is easy to estimate the derivatives $I_x$, $I_y$, and $I_t$ directly from the image.

Equation (2.30) can be represented, as shown in Figure *2.6*. The Figure shows that the values of u and v that satisfy the equations are lying on the straight line as shown.

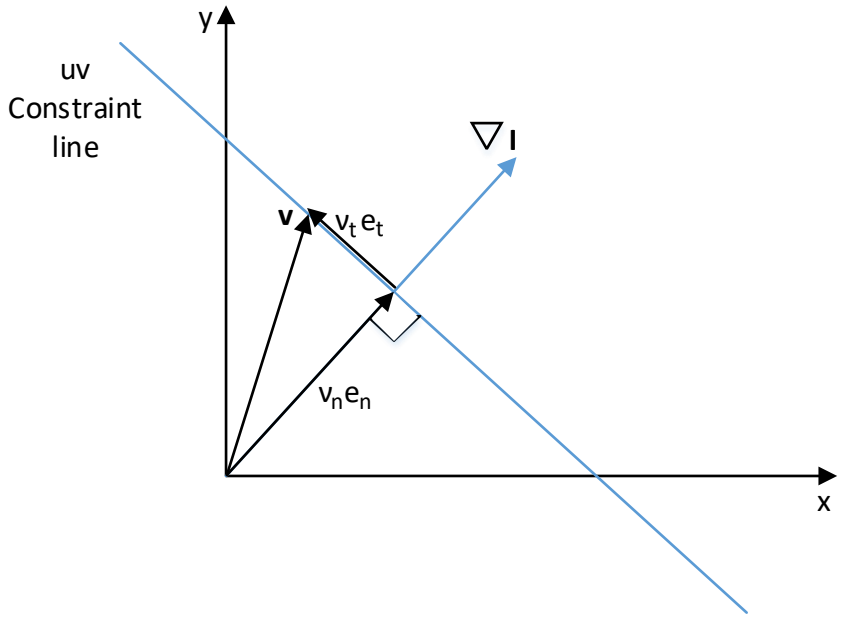



Figure 2.6: $uv$ constraint line. The motion vector (**v**) decomposition into normal ($v_n\ e_n$) and the tangent ($v_t\ e_t$) components. Any tangent component can satisfy the optical flow equation for given $\nabla I$ and $I_t$.

Equation (2.30), however, is ill-pose as it is one equation with two unknowns $u$ and $v$. All vectors parallel to the constraint line described by Equation (2.30) are likely to represent the real estimation of the optical flow.

From Figure 2.6, the flow vector can also be represented as:

$$\boldsymbol{v} = v_n \boldsymbol{e}_n + v_t \boldsymbol{e}_t \tag{2.32}$$

here, $\boldsymbol{e}_n$ is in the normal vector in the direction of the image gradient $\nabla\mathbf{I}$, while $\boldsymbol{e}_t$is orthogonal to $\boldsymbol{e}_n$.

Then the optical flow (Equation (2.30)) can be rewritten as:

$$v_n \|\nabla \boldsymbol{I}\| + I_t = 0 \tag{2.33}$$

where $\|\nabla \boldsymbol{I}\|$ is the magnitude of the gradient vector (Yao Wang, 2001).

The optical flow at right angles to this direction is ambiguous and cannot be calculated. This phenomenon is usually called the aperture problem. Equation (2.30) is formulated for single-point optical flow estimation, and a discrete realization of the spatial and temporal derivatives can be acquired within some neighbourhood of the image point. In Figure 2.7(a), if the spatial gradient shows directional derivatives within this area, the optical flow can be estimated unambiguously. The image

has gradients in two different directions since the constraint lines of several points within the neighbourhood will intersect in one point. On the other hand, in Figure 2.7(b), it is impossible to determine whether the motion is upward or perpendicular to the edge because there is only one gradient direction in the aperture. Besides, the same ambiguity arises when the image area is textureless Figure *2.7*(c).

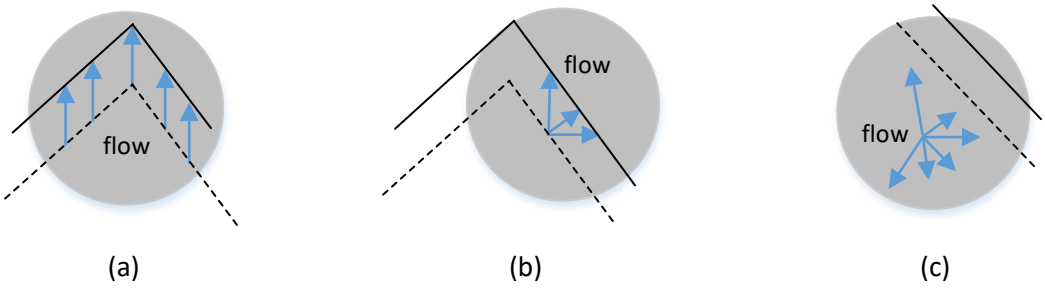



Figure 2.7: The effect of the area structure on the optical flow estimation. (a) no aperture problem with moving corners (spatially distributed structure). (b) The case of moving edge where the local neighborhood of all the gradients are parallel. In this case, the estimation of the optical flow in ambiguous. (c) in the case of a textureless surface, the constraint becomes completely inapplicable *(Szeliski, 2011)*.

Clearly, one constraint is not enough to calculate the optical flow. Thus, other constraints are often introduced, like motion smoothness. The motion field varies smoothly across the image. For instance, the moving object's projection results in a motion field with an equal direction across the pixels that represent it. Thus, the second constraint can be obtained from the variational calculus by first minimizing a measure of departure from smoothness as:

$$e_s = \iint \left( \left(u_x^2 + u_y^2\right) + \left(v_x^2 + v_y^2\right) \right) dxdy \tag{2.34}$$

The optical flow constraint error:

$$e_c = \iint (I_x u + I_y v + I_t)^2 dxdy \tag{2.35}$$

should also be small.

The task is to minimize $e_s + \gamma e_c$ , where $\gamma$ is a weighting parameter for the image motion relative to the departure from smoothness. If intensity measurement is accurate $\gamma$ will be high, otherwise small when there is noise.

The integral of the form of Equations (2.35) is a minimization of the function:

$$\iint F(u, v, u_x, u_y, v_x, v_y) dxdy \tag{2.36}$$

corresponding to Euler equations of the form:

$$F_u - \frac{\partial}{\partial x}F_{u_x} - \frac{\partial}{\partial y}F_{u_y} = 0 \tag{2.37}$$

$$F_v - \frac{\partial}{\partial x}F_{v_x} - \frac{\partial}{\partial y}F_{v_y} = 0 \tag{2.38}$$

In this case,

$$F = \left(u_x^2 + u_y^2\right) + \left(v_x^2 + v_y^2\right) + \lambda\left(I_x u + I_y v + I_t\right)^2 \tag{2.39}$$

so, the Euler equations yield:

$$\nabla^2 u = \lambda(I_x u + I_y v + I_t)I_x \tag{2.40}$$

$$\nabla^2 v = \lambda(I_x u + I_y v + I_t)I_y \tag{2.41}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian operator (Kroeger, 2016; Zach, 2007).

## 2.8 Depth from Stereo Image

One of the important tasks in computer vision is to calculate the distance of an object or a point with respect to the camera. From cameras with a known displacement between them in stereo settings, the depth information can be retrieved utilizing image intensity. This particular case of binocular stereo geometry depicted in Figure 2.8. In this simple setup, the cameras are separated by a baseline distance $T$ in the $x$ direction. The image planes are on the same plan or coplanar in this arrangement. Generally, the stereo camera has essential configuration properties that allow better resolution of disparity estimation as the baseline T between the two images increases. However, the coplanar setup introduces another problem if T increases because the corresponding points between the two scenes will decrease.

Furthermore, the perspective projection introduces distortion between the region of correspondence. However, it will be seen later that this coplanar configuration is the target configuration to obtain the image correspondence in the stereo setup. The correspondence between pixels is utilized to estimate disparity. Disparity represents the displacement between the locations of the feature projection on the two image planes. The camera centers, and the object points construct the epipolar plane. The epipolar plane intersects with the image planes at the epipolar line. In the model of Figure 2.8, the projection of the object point on both image planes will be on the same epipolar line of the two images (Oram, 2001).

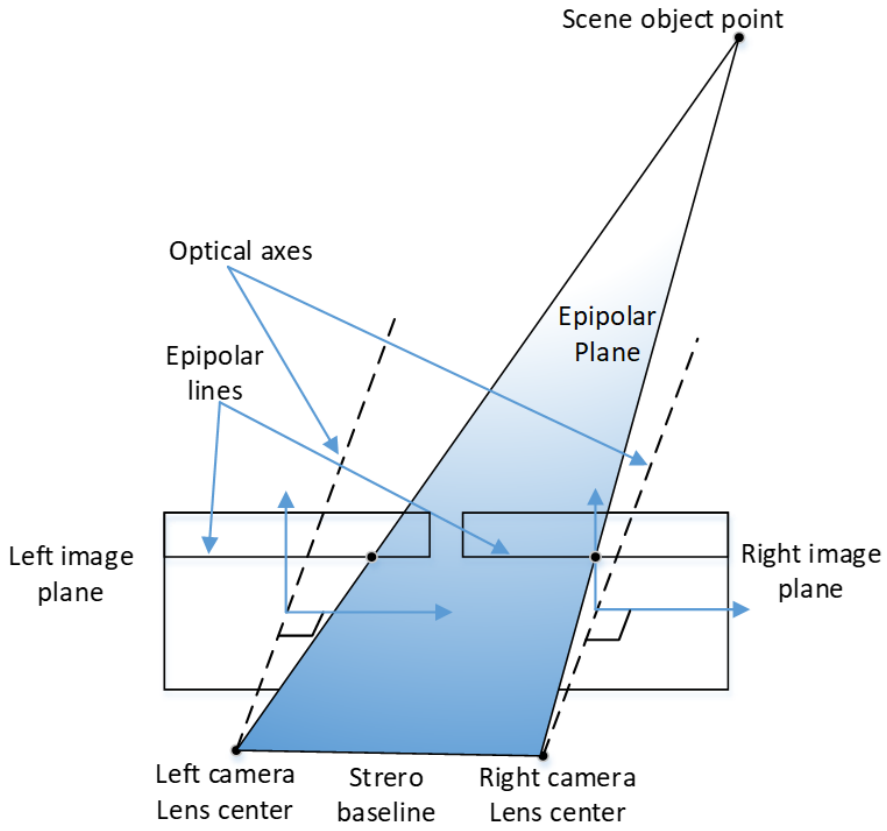



Figure 2.8: Binocular stereo geometry

To calculate the disparity, the epipolar line must be the same in both images, and the search for the pixel correspondence is performed at that line.

In Figure 2.9 the same binocular stereo geometry is shown. Both points $p_t$ and $p_r$ represent conjugate pair since they are the projection of the same point ($P$). If the two images are superimposed on each other, the distance between the two conjugate pairs is the disparity. Now, if the coordinate of the origin is coinciding with the left camera center, the triangles $PMC_l$ and $p_lLC_l$ are similar. Thus,

$$\frac{x}{z} = \frac{x'_l}{f_L} \tag{2.42}$$

In the same way, $PNC_r$ and $p_rRC_r$ are similar, then:

$$\frac{x-b}{z} = \frac{x'_l}{f_L} \tag{2.43}$$

And from both equations, the depth value z can be estimated as:

$$z = \frac{bf_L}{(x'_l - x'_r)} \tag{2.44}$$

Therefore, the scene depth can be estimated by knowing the disparity map of the scene and applying the equation above, where both $b$ and $f$ are the camera's constant parameters.

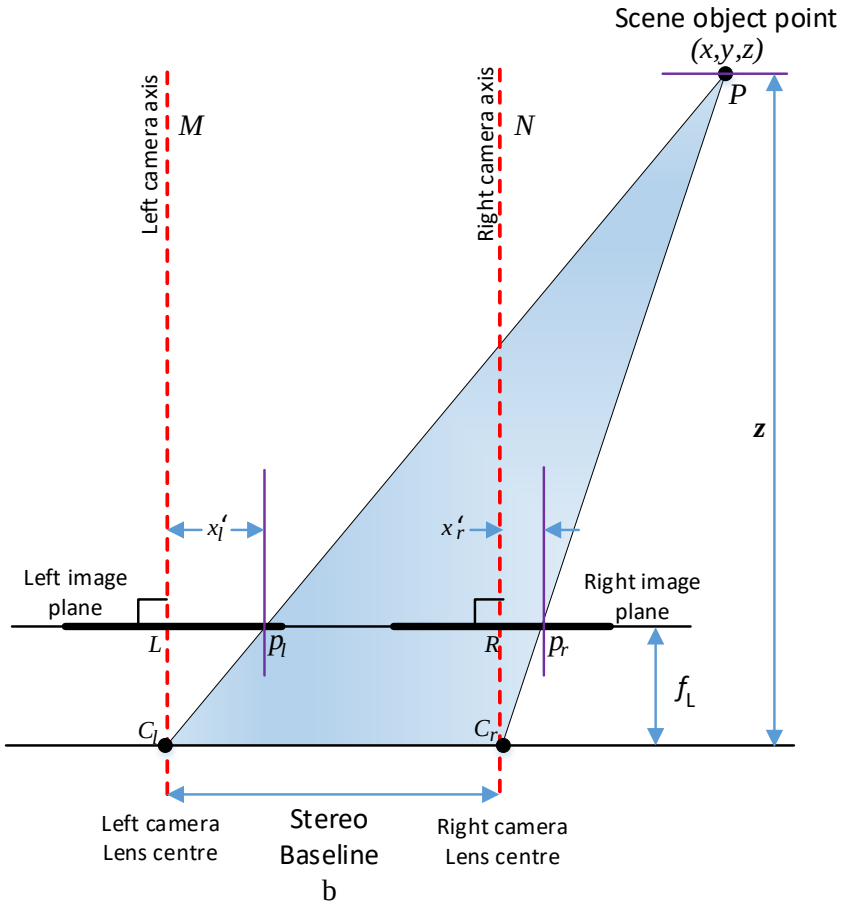



Figure 2.9: Depth Triangulation Estimation using the triangular similarity. Both triangles $PMC_l$ and $p_lLC_l$ are similar as well as $PNC_r$ and $p_rRC_r$.

Two camera setup involves two steps:

a) Mathematically remove radial and tangential lens distortion, which is also called undistortion.
b) Adjustment for angles and distances between cameras, in what is called rectification. This process leads to coplanar image planes, and the corresponding rows are colinear.

From Equation (2.44) above, the relation between the disparity and the depth is nonlinear since the disparity is inversely proportional to depth. Figure 2.10 shows that a small disparity difference makes a large depth difference when the disparity is near zero. On the other hand, when the disparity is large, a small disparity difference will not affect the depth value in a profound value. Thus, the depth estimation near the camera will be more accurate as the object is closer to the camera system.

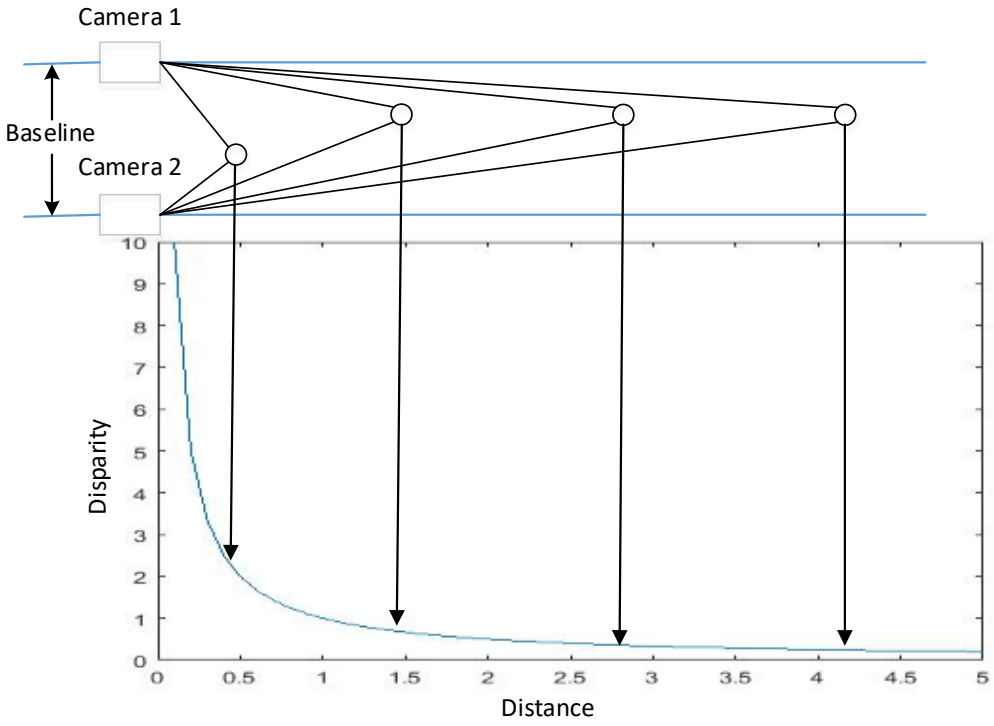



Figure 2.10: Disparity-Depth relationship in parallel image plane setup of Figure 2.9.

Figure 2.10 is a simplified structure of parallel image planes that are not practical to implement. There will be no disparity value of zero that should reflect the vanishing points (image point at infinity).

The more realistic binocular implementation is the converged setup, which means a slight tilting of the image plane inside, allowing convergent or vanishing point to exist that should result in a zero disparity value. Figure *2.11* shows that more realistic implementation. With this setup, the relationship between the disparity and depth cannot be as straight as in Equation (2.44), and other issues need to be considered.

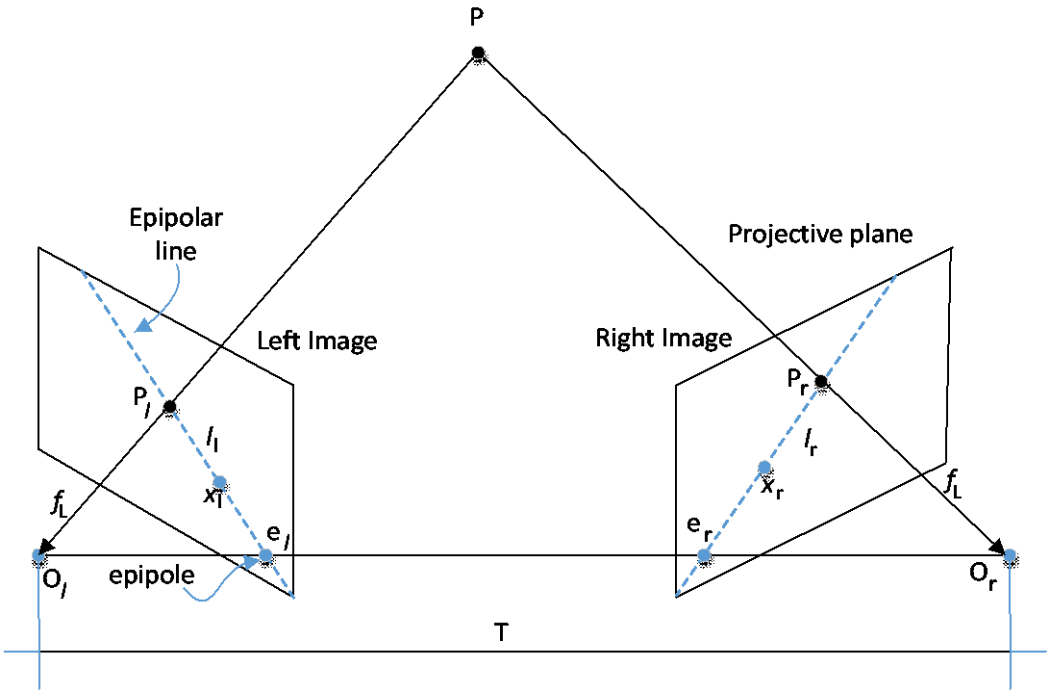



Figure 2.11: Converged binocular camera setup.

Having the two image planes tilted to each other will produce the following configuration as in Figure 2.11. The geometry is referred to as the epipolar geometry. In fact, this model combines two pinhole models and utilizes the epipolar line in the way that is shown. Now, there are a new definition for epipoles, $e_l$ and $e_r$. Here, $e_l$ ($e_r$) is the projection of the image center $O_l$ ($O_r$) on the image plane and aligned with $O_r$ ($O_l$). Thus, the epipolar line is the line that can be formed between the projection of the point $P_l'$ ($P_r'$) and the epipole $e_l$ ($e_r$).

Epipoles geometry provides an essential clue in estimating the image disparity or depth in comparison to single pinhole camera geometry. It can be seen from Figure (2.13) that if we consider only the left camera, the point P could be anywhere along the line $PO_l$ and no extra information will be available about how far the point from the image plane is. The possible locations of point P along the path $PO_l$ could be anywhere. However, all the possible locations of $P$ on $PO_l$ are projected on the epipolar line $P_r e_r$. The same can be said about all the possible locations of the point P along the path $PO_r$ with their projection on $P_l e_l$. This case tells that searching for the corresponding point between stereo images must be located on the epipolar lines. This constraint is usually called the epipolar constraint, and it simplifies the searching of the corresponding points between two images.

To this end, the relation between $P_r$ and $P_l$ which represents the physical projection of the point $P$ on the right and left image could be obtained as follows:

From Figure *2.11*, the point $P_r$ can be represented as:

$$\overrightarrow{P_r} = R.\left(\overrightarrow{P_l} - \vec{T}\right) \tag{2.45}$$

Now, considering points $\vec{x}$ belong to a plane with a normal vector $\vec{n}$ and passing through a point $\vec{a}$ that obeys $(\vec{x} - \vec{a}) \cdot \vec{n} = 0$. Then a relation between $\overrightarrow{P_l}$ and $\vec{T}$ can be deduced as:

$$\left(\overrightarrow{P_l} - \vec{T}\right)^T \left(\overrightarrow{P_l} \times \vec{T}\right) = 0 \tag{2.46}$$

From Equation (2.45):

$$\left(\overrightarrow{P_l} - \vec{T}\right) = R^{-1} \cdot \overrightarrow{P_r} = R^T \cdot \overrightarrow{P_r} \tag{2.47}$$

thus

$$\left(R^T \cdot \overrightarrow{P_r}\right)^T \left(\overrightarrow{P_l} \times \vec{T}\right) = 0 \tag{2.48}$$

The cross product can be replaced by a matrix multiplication (Szeliski, 2011) as:

$$\overrightarrow{P_l} \times \vec{T} = S.\overrightarrow{P_l} \tag{2.49}$$

where $S$ can be expressed as:

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{2.50}$$

and Equation (2.48) can be rewritten as:

$$\overrightarrow{P_r^T} \cdot R \cdot S \cdot \overrightarrow{P_l} = 0 \tag{2.51}$$

or:

$$\overrightarrow{P_r^T} \cdot E \cdot \overrightarrow{P_l} = 0 \tag{2.52}$$

here, $E$ called the essential matrix and it relates the geometry between the projection of the point P on the image plane and it has no relation to the final projection of the point P on the image plane. It

provides the two cameras' geometry relative to each other but without information about the camera's specifications. Nevertheless, we can still relate the project point $P_r$ and $P_l$ on the image plane in terms of the $p_r$ and $p_l$ using Equation (2.52) to obtain:

$$\overrightarrow{p_r}^T \cdot E \cdot \overrightarrow{p_l} = 0 \tag{2.53}$$

Note that the essential matrix $E$ maps a point $x_l$ in the left image into a line $l_r = Ex_l$ in the right image, since $x_l l_l = 0$ and the same applied to $l_l$. Both $l_r$ and $l_l$ are called the Epipolar lines (Kaehler & Bradski, 2016).

The intrinsic matrix $K$ from Equation (2.11) needs to be involved to obtain the pixel coordinates. Then, Equation 3.52 can be rewritten as:

$$\overrightarrow{p'_r}^T \cdot (K_r^{-1})^T \cdot E \cdot K_r^{-1} \cdot \overrightarrow{p'_l}^T = 0 \tag{2.54}$$

where $\overrightarrow{p'_r} = K.\overrightarrow{P_r}$ and $\overrightarrow{p'_l} = K.\overrightarrow{P_l}$

The fundamental matrix is defined as:

$$F = (K_r^{-1})^T \cdot E \cdot K_r^{-1} \tag{2.55}$$

and thus:

$$\overrightarrow{p'_r}^T \cdot F \cdot \overrightarrow{p'_l}^T = 0 \tag{2.56}$$

So the difference between the Fundamental and the essential matrices is that the fundamental matrix works directly on the image pixel. In contrast, the essential matrix operates on the physical geometry of the image.

## 2.9 Stereo Calibration

For a single image, calibration retrieves the pixel/meter relationship. In a stereo image setting, calibration is more important because it is a constraint in estimating the pixel disparity or depth from the stereo image. This constraint states that pixels need to be aligned horizontally such that each pixel has the same y-coordinate spatially with its corresponding pixel in the second image. It can be achieved by stereo rectification, which requires calibration as a first step. In concept, calibration removes any distortion in the image due to non-linearity in the camera lenses. In the case of the stereo image, each image will have different nonlinear characteristics because they belong to two different cameras. Stereo calibration is the process of computing the geometrical relationship between two

cameras in space. Calibration mainly depends on the rotation matrix R and translation vector T between the two cameras. In general, a point $P$ in 3D can be represented from one camera in terms of the other camera as:

$$\overrightarrow{P_l} = R_l \cdot \vec{P} + \vec{T} \text{ and } \overrightarrow{P_r} = R_r \cdot \vec{P} + \vec{T} \quad (2.57)$$

Also from Figure *2.11* that the left projection $\overrightarrow{P_l}$ can be represented in terms of the $\overrightarrow{P_r}$ as:

$$\overrightarrow{P_l} = R^T \cdot \left(\overrightarrow{P_r} - \vec{T}\right) \quad (2.58)$$

Solving Equations (2.57-2.58) for $R$ and $\vec{T}$ yields:

$$R = R_r \cdot R_l \quad (2.59)$$

$$\vec{T} = \overrightarrow{T_r} - R \cdot \overrightarrow{T_l} \quad (2.60)$$

In this project, the dataset utilized for training and testing for disparity estimation was already stereo calibrated, so no further processing is needed. However, for some qualitative results, the equations above have been utilized to calibrate stereo image pairs before e feeding them to the DL model.

## 2.10 Disparity Estimation

Both disparity and optical flow follow the same mathematical model in estimation; however, the disparity is more related to the object's position while the optical flow is related to the project 2D motion of the object on the image plane.

Let's denote the sequence of images as $I: \Omega \,.\, \tau \rightarrow \mathbb{R}$ , where $\Omega \rightarrow \mathbb{R}^2$ is the image domain and $\tau$ is the frame spacing. The primary assumption in estimating the optical flow is that the pixel intensity stays constant during the displacement. Thus, the brightness constancy constrained can be defined as:

$$\frac{dI}{dt}(\boldsymbol{x}(t), t) = 0 \quad (2.61)$$

In the discrete domain, the equation above can be approximated for a giving pixel $x \in \Omega$ and as:

$$I(\boldsymbol{x} + \boldsymbol{w}(\boldsymbol{x}), t+1) - I(\boldsymbol{x}, t) = 0 \quad (2.62)$$

Equation (2.62) can be simplified to be in a more appropriate form for optimization algorithms using partial derivatives that result in a linear version.

$$\frac{\partial I}{\partial x}(\boldsymbol{x})u(\boldsymbol{x}) + \frac{\partial I}{\partial y}(\boldsymbol{x})v(\boldsymbol{x}) + \frac{\partial I}{\partial t}(\boldsymbol{x}) = 0 \tag{2.63}$$

Equation (2.63) can also be written in compact form as:

$$\nabla I(\boldsymbol{x}).w(\boldsymbol{x}) + I_t(\boldsymbol{x}) = 0 \tag{2.64}$$

The $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$ is the spatial gradient operator, and $I_t(\boldsymbol{x}) = \frac{\partial I}{\partial t}$ is the partial derivative over the image pair.

The brightness constancy constraint of Equation (2.63) contains two unknown $(u(\boldsymbol{x}), v(\boldsymbol{x}))$ motion components. In this Equation, the image gradient multiplication with the motion components can generate what is known as the aperture problem, which is the ambiguous motion with respect to the neighboring context.

It is worth to mention here that Equation (2.63) is an optical flow equation. Still, this projection is also exploited as a disparity estimation equation by considering that there is no change in the motion in the y-direction. The equation satisfies the epipolar constraint. The model needs to find the corresponding pixel on the same line between the stereo images.

The projection of the object moving in the 3D on the image plane is usually referred to as optical flow. It is also referred to as this projected motion as the apparent motion because it only represents image 2D. One of the well-known problems with the optical flow is the case of a sphere or cylinder illuminated with a uniform and non-textured surface rotating around an axis parallel to the image plane. In this case, estimating optical flow will give no apparent motion for the projected pixels from the object on the image plane. On the contrary, if a static object is illuminated with a constantly moving source of light, it will produce optical flow representation. These problems depend on how it is calculated by means of intensity constancy assumption of the object within the image. Optical flow is mainly employed in object tracking in addition to medical registration. Furthermore, it can operate on video frames without the need for rectification, as in the disparity or depth estimation. Lucas-Kanade and Horn-Schunck addressed the optical flow estimation problem in two distinctive methods (Baker & Matthews, 2004).

On the other hand, the disparity is surrogate to depth estimation. Using feature correspondence, the displacement between the feature position in the left and right image is defined as the disparity shown in the figure below. As explained, the Epipolar line between the two images should be the same, and the point should be located within this line in both images, which implies that the two images are rectified. The maximum disparity parameter is used to restrict the searching distance to a specific limit. Generally, Equation (2.63) can be applied in the same situation with the condition that $\partial y$=0. The usage of disparity is mainly to calculate the depth and related topics like 3D object reconstruction.

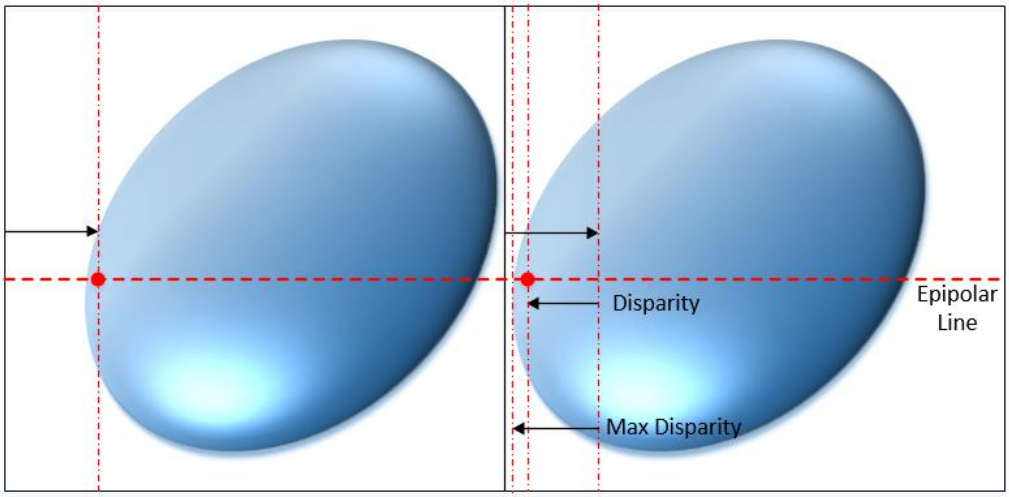



Figure 2.12: Disparity Estimation in binocular image set. The Max Disparity represents the maximum searching space for the disparity algorithm to find the disparity value.

## 2.11 Occlusion Problem

The multi-image settings experiencing different types of problems, nonrigidity, transparencies, reflections, and lack of texture are a few of these problems that computer vision addresses. However, depth discontinuities or occlusion is dominating as an essential issue in depth and disparity estimation, and it appears in almost all image models. Figure 2.13 shows the occlusion case of the object when projected in stereo images. It causes the object to appear and disappear between the two viewpoints. In this situation, the occluded pixels in one image will not have any correspondence in the second image, which introduces ambiguities to the motion estimation process. These occluded pixels cause the motion estimation to not be observable at these pixels.

In the previously mentioned methods of disparity estimation, no care has been taken regarding the pixels without correspondence. This issue is especially important when the requirement is to render a virtual or 3D view where all the pixel disparity needs to be known. The same problem is rising when video frame rate conversion is required, where the motion of all pixels is required to operate. Hence, a careful consideration to estimate the occlusion issue can improve the motion estimation of the

disparity and other motion estimation techniques. The occlusion estimation's importance can be noticed on a large motion displacement of the object where the object is close to the camera in stereovision or similar motion estimation like optical flow. Therefore, the image needs to be segmented to occluded and nonoccluded pixels and treat the occluded pixels differently when estimating the disparity.

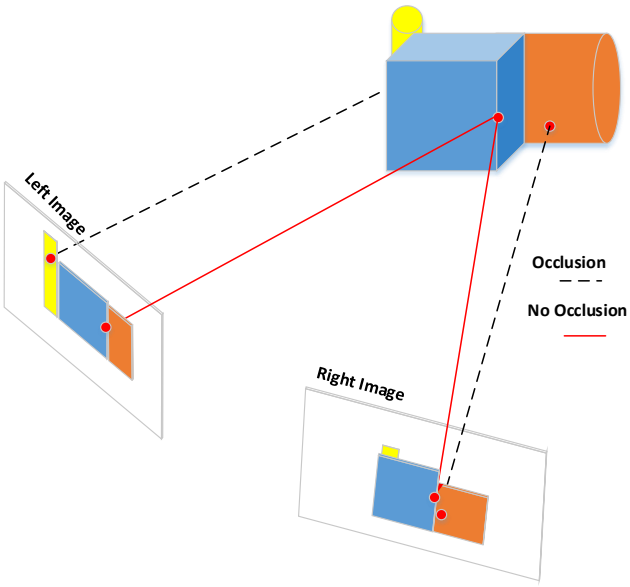



Figure 2.13: Object Occlusions. The orange cylinder on the right is shown wider in the right image while smaller on the left image. In the other case, the yellow cylinder is entirely shown on the left image while it disappears on the right image and only shows on top of the blue object.

Occlusion detection is a kind of problem with whom to start first. Figure 2.14 below shows that both areas A and B represent occlusion concerning the image facing them. It means that pixels in the A range on the image $I_R$ has no corresponding pixel in the image $I_L$. Similarly, for B area. To clarify it further, assume that we what to render another image *J* between $I_L$ and $I_R$. In a simple situation, the disparity between the two images should be calculated and then compensated to produce *J*. However, since both areas A and B are experiencing occlusion, it is impossible to calculate the occlusion at the areas $O_L$ and $O_R$ that are projected on both $I_R$ and $I_L$, respectively.

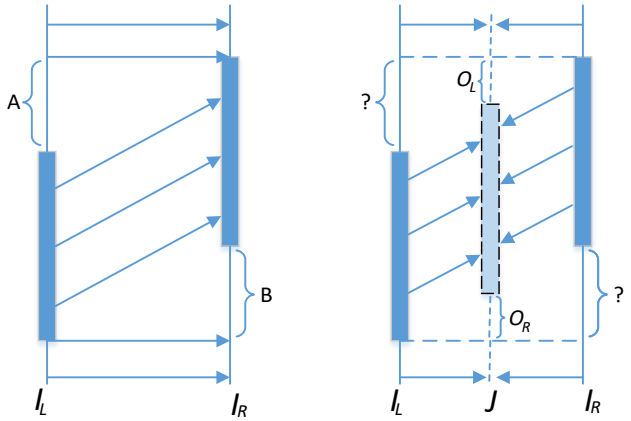



Figure 2.14: Occlusion detection. To render a new image, J from the left and the right image information from A and B areas need to be available to reconstruct the OL and OR areas. It is impossible to do that without incorporating the two disparity maps *(Ince & Konrad, 2008)*.

However, it is possible to use the disparities of both images to estimate the occluded area disparity. Figure 2.15 shows image pairs with their corresponding disparities and the hidden pixels due to occlusion for the two images. Estimation of the occlusion pixels is performed by estimating Equations (2.65,2.66) (Ince & Konrad, 2008):

$$DL_{occ} = dispL(\boldsymbol{x}) + dispR\big(\boldsymbol{x} + dispL(\boldsymbol{x})\big) \tag{2.65}$$

$$DR_{occ} = dispR(\boldsymbol{x}) + dispL\big(\boldsymbol{x} + dispR(\boldsymbol{x})\big) \tag{2.66}$$

Thus, the occlusion detection can be performed after estimating the disparity from both images (left and right) and applying the above two equations. Note that the left disparity contains a matrix of negative values, and this means the resultant of the above Equations (2.65,2.66) are usually values of values between 0 and 1. The last pair of Figure 2.15 shows such occlusion estimation, with the dark areas representing the occlusion. The green areas represent zero values which is usually the area where correspondence exists between pixels.

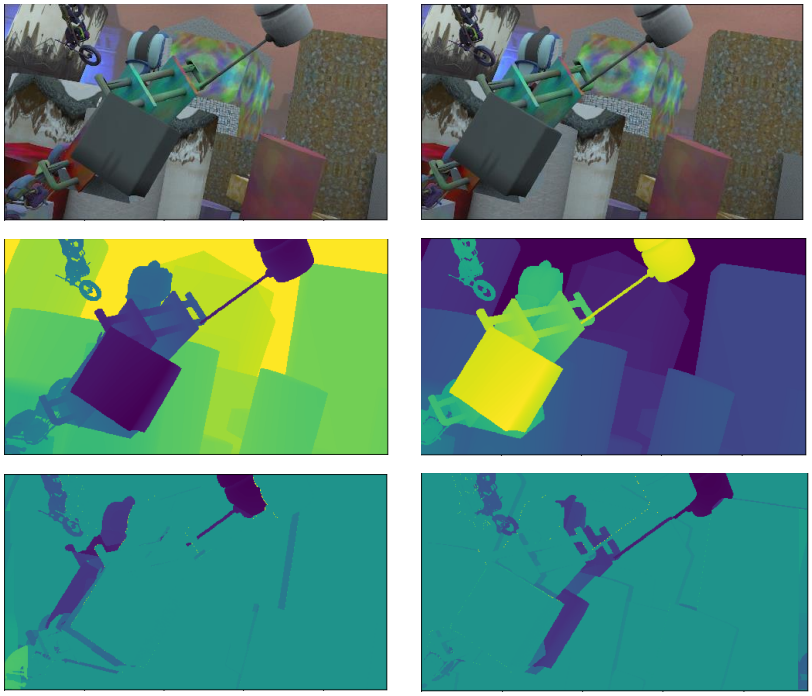


Figure 2.15: FlyingThig3D Dataset. Top stereo: image pairs *(Mayer, et al., 2016)*. Middle: Disparities of the left image (negative values) and the right image. Down: occlusion of the left and right images, respectively. Clearly, in the occlusion areas, the pixels within this area have no correspondence.

## 2.12 Stereo Rectification

After the calibration, image rectification is required to estimate the image disparity for the stereo image. Image rectification is the process of adjusting the stereo image rows to be aligned horizontally. This process implies that the two image planes must be aligned precisely or referred to as coplanar, which allows finding pixels on the same row of the second image with some displacement representing the disparity. The result of such alignment is equivalent means that the epipoles are located at infinity. Examples of the available algorithms that perform rectification include Harley's algorithm (Chen, et al., 2003) and Bouguet's algorithm (Kim & Kim, 2013).

After these processes, the images can be utilized to estimate the disparity of the image. This process of algorithms allows many other applications that can be implemented to perform other types of application like 3D reconstruction, for example. Here the 3D reconstruction can be estimated utilizing what is referred to as reprojection matrix (Laskar, et al., 2015):

$$Q = \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f_L \\ 0 & 0 & -\frac{1}{T_x} & \frac{c_x - c'_x}{T_x} \end{pmatrix} \tag{2.67}$$

where $(c_x, c_y)$ is the principal point coordinate in the left image and $c'_x$ is the $x$-coordinate of the principal point in the right image. Considering the intersection of the principal point at the infinity, then $c_x = c'_x$, the lower-right term in the equation tends to zero. $T_x$ is the baseline between the camera's centres. Using the $Q$ matrix above, the reprojection of the image coordinates to the 3D coordinate can be obtained as:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ d(x,y) \\ 1 \end{bmatrix} \tag{2.68}$$

here, $d(x,y)$ is the disparity at the point $(x,y)$, and the 3D coordinates are then $(X/W, Y/W, Z/W)$, Figure *2.16* shows an image in 2D and its real dimension reconstructed in 3D.

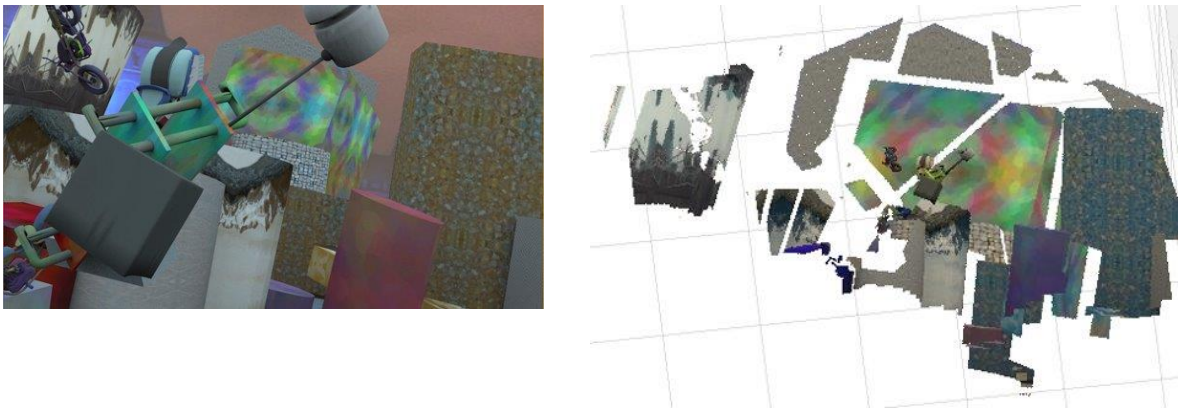


Figure 2.16:3D Scene Reconstruction. Using the disparity estimation in companion with the camera specification to reconstruct the 3D scene from single image, the left image obtained from FlyingThings3D dataset *(Mayer, et al., 2016)*.

There is an alternation between optical flow and correspondence or disparity concerning the apparent motion during this chapter. Both share almost the same concept or notion that motion can be inferred from image or video, and they are sharing the same mathematical concept. However, the applications related to these concepts differ. Traffic analysis and object tracking are well-known applications that mainly utilize optical flow. While 3D reconstruction for augmented reality is usually associated with disparity estimation. In the meantime, both can be exploited in segmentation (Mikic, et al., 1998) (Aach & Kaup, 1994)

## 2.13 3D Displacement Field

Unlike the stereo image problem, the 3D motion of an object (rigid or non-rigid) is assumed to be not restricted with rectification, as in the disparity estimation. For this case, the transformation is free in the three dimensions. Furthermore, the 3D displacement field is considered a free and arbitrary

relative motion that does not suffer from the implication of the apparent motion definition as in the optical flow. In the camera model of Figure 2.17, the absolute or the relative motion of a point in 3D can be estimated by the motion of the projected point **P** as shown. As depicted, the 3D motion can be determined by estimating the projected position of point **p.**

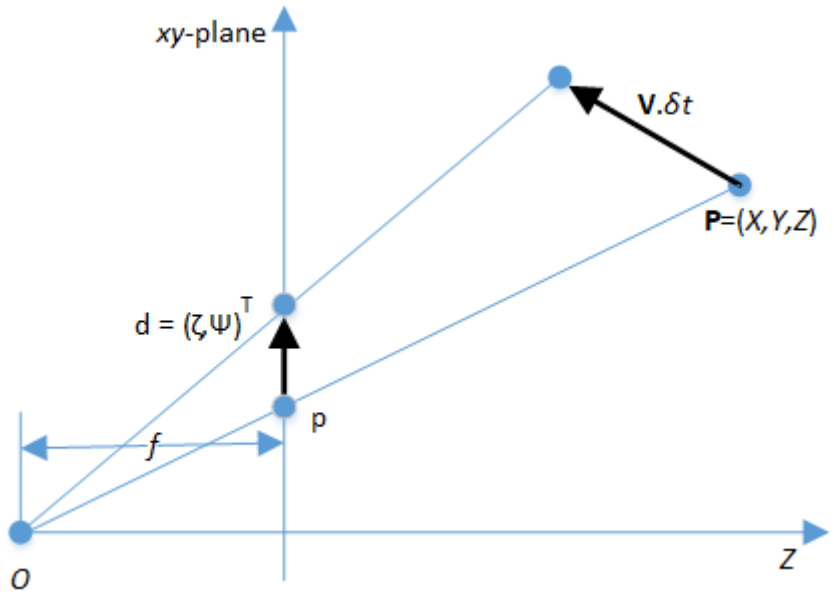



Figure 2.17: Displacement Field

## 2.14 Summary

Motion analysis in video and multi-imaging systems addresses many vital topics in today's research activities and provides solutions for different fundamental problems in computer vision. Motion detection and classification are mainly action recognition tasks that consider the sequence of motions within a video and the ambient during this motion. Detection of a specific motion within an input containing large variations in the illumination and object appearance is a challenging task, especially when the correspondence across input frames is required to be maintained. Motion analysis involves other types of applications like optical flow, disparity, segmentation, and other topics. The main challenge in this field is to implement a dense representation that can provide the required information at each pixel of the image. The camera model is essential in estimating the projection of the 3D scene on the image plane, and it is also used to reconstruct the 3D scene from the image.

In this chapter, two models of motion presented the projection and the apparent motion. The estimation of each model is different from the other. While the prior depends on the camera parameters and settings. The latter depends on the fact that the illumination between the frames is not changing and this constraint with other constraints is utilized to estimate the motion in the apparent motion.

# Chapter 3 - DEEP LEARNING CONCEPTS

## 3.1 Introduction

The progress in machine learning, particularly in the neural network (NN), experienced many ups and downs. The research in this field was accelerating and flourish, then dies, then start again, and so on. After developing the perceptron by F. Rosenblatt in 1958 for a classification task of linearly separable data, the researchers abandoned it when they found in 1969 that it suffers a severe drawback. The model could not perform XOR function, which is a nonlinear operation. Although it had been found out that this same function can be performed using multiple layers of the perceptron, they faced another problem of training a model with multiple layers till 1986. In that year (Rumelhart, et al., 1986) suggested the backpropagation training methods to train multiple layers of perceptron models. This new finding injected new life into the NN field, and since then, a tremendous amount of research and advancement has been achieved.

Another advancement was introduced by adding an activation function to the perceptron to improve the model's performance for nonlinear problems. But the demand for increasing the network's power by increasing its number of layers to approximate more complex functions was still a problem. Increasing the network size triggers vanishing gradient and other generalizations issues when the number of layers increases to more than four. For these reasons, the research in the field again slows down. However, with the deep convolutional neural network (LeCun, et al., 1998), the attention returned to this technology. The new findings overcome many of the previous problems that the previous NN models suffer, especially the vanishing gradient problem. Since then, researchers taking an extensive effort in the field, and by 2012 large model trained on a GPUs (Krizhevsky, et al., 2012) was born. The advancement in this field is ongoing, new software and hardware are developing especially for this technology. In software, deep learning frameworks like Tensorflow, Pytorch, Caffee, and many others are drawing the researcher's attention. On the hardware side, new optical GPUs are developed to increase the speed of the current GPUs.

In this chapter, the neural network and its main components will be discussed, then the importance of the regularization and the different techniques to achieve it will be reviewed. The optimizers and their usage in training will be presented. Furthermore, various well-known networks architectures will

be outlined. Then the explanation of the specialized networks like Spatial Transformer Network (STN), Sequence networks, and Generative Adversarial Network (GAN) will be reviewed.

## 3.2 Feed-Forward Neural Networks

The human brain consists of highly structured connections called a synopsis, which consists of simple elements called neurons. With the human sensory system's help, the brain can perform very complicated tasks with high efficiency. The artificial neural network mimics this highly efficient structure that learns from observation and saves what it learns in layers of features constructed from these simple repetitive elements of neurons.

### 3.2.1 Feed-Forward Multi-Layer Neural Networks

In the multi-layer neural network, there is more than one layer of neurons. The neurons are arranged such that each element that belongs to the input vector is connected to all the neurons in the input layer, and each neuron in the input layer is connected to all the neurons in the hidden layer. This type of connection arrangement is called a fully connected network. In the meantime, there are no connections between the neuron units within the same layer. In a multi-layer perceptron, the mapping function from the input $x$ to the output $y$ can be defined as:

$$y_k(\boldsymbol{x};\boldsymbol{\theta}) = f^2\left(\sum_{j=1}^{M} W_{kj}^{(2)} f^1\left(\sum_{i=1}^{D} W_{ji}^{(1)} x_i\right)\right) \tag{3.1}$$

here M and D represent the hidden and the input number of units. The superscripts represent the layer index and $\boldsymbol{\theta} = \{\boldsymbol{W}^{(1)} \in \mathbb{R}^{M\times D},\ \boldsymbol{W}^{(2)} \in \mathbb{R}^{K\times M}\}$ is the parameter set.

Equation (3.1) arrangement can be applied multiple times to construct multiple layers using the same equation. More importantly, the equation's structure can collapse to one layer if implemented without an activation layer. Conventionally, between any consecutive layers, an activation function is usually applied that introduces nonlinearity to the model. Activation functions like a sigmoid, hyperbolic tangent, rectified linear unit (ReLU) are among many activation functions that can be utilized to introduce the nonlinearity property within the model.

Giving the training set $\{\boldsymbol{x}_n, \boldsymbol{t}_n\}_{n=1}^{N}$ as in a classification task, the aim of Equation (3.1) is to find the set of parameters $\boldsymbol{\theta}$ that achieve the best mapping of $\boldsymbol{x}$ to $y$. The common cross-entropy cost function can be defined as:

$$E(\boldsymbol{\theta}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log\left(y_{nk}\right) \tag{3.2}$$

where $t_{nk}$ is the $k$th one hot vector style of the target vector $t_n$ and $y_{nk}$ is the $k$th prediction of the $y_n$ when applying $x_n$ at the input. Now, finding $\theta$ that minimizes the error in Equation (3.2) is not solvable for its highly non-linearity and dimensionality in a non-convex representation. Activation Function

A nonlinearity operation is usually applied after each layer. The nonlinearity function enhances the model with more expressive power compared to the linear model. The activation function needs to have specific characteristics to be considered for the training process. The function needs to be monotonically increasing function, differentiable, and continuous (Hornik, 1991).

#### 3.2.1.1 Sigmoid and Hyperbolic Tangent

Sigmoid and hyperbolic tangents are among the oldest and still commonly used activation functions. For instance, the sigmoid complies with the requirement of the activation function characteristics required. It is also bounded between [0,1] over the range of the input with the form expressed as:

$$f(x) = \frac{1}{1 + e^{-\beta x}} \tag{3.3}$$

The value of $\beta$ controls the steepness of the function curve near the origin. The derivative of the sigmoid function is shown in Equation (3.3)is:

$$f'(x) = \beta f(x)(1 - f(x)) \tag{3.4}$$

which is easy to implement, and it approaches zero as $x$ moves away from zero.

The hyperbolic tangent has quite similar characteristics, but it is bounded between [-1,1] over the input range in addition to the required attributes for a successful training process of a model. The hyperbolic tangent can be expressed by the symmetrical form of Equation (3.5):

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.5}$$

The steepness of the function across the origin can be controlled by applying $\beta x$ instead of $x$ to the function, and similarly, its derivative will be symmetrical with a max value of $\beta$ as shown in Equation (3.6).

$$f'(x) = \beta(1 - f(x)^2) \tag{3.6}$$

However, it has been found that these functions are causing a problem during the training of the deep model.

#### 3.2.1.2 Rectified Linear Unit

Researches showed that other, more straightforward activation function, like Rectified Linear Unit (ReLU), can overcome the issues that sigmoid and hyperbolic tangent suffers, (Nair & Hinton, 2010). ReLU can be defined as:

$$f(x) = max(0, x) \tag{3.7}$$

and the gradient of ReLU function that can be defined as:

$$f'(x) = \begin{cases} 0 & if\ x \leq 0 \\ 1 & if\ x > 0 \end{cases} \tag{3.8}$$

proved that it could improve the training speed (Krizhevsky, et al., 2012).

The problem that the ReLU came to address are two, first is the vanishing gradient that the Sigmoid a Hyperbolic Tangent functions cause. For instance, in the case of these functions, when the derivatives are smaller than one and multiplied by each other, the multiplication will become smaller, and the gradient starts to tend toward zero and vanish. On the other hand, if the derivatives are greater than one, the resulting multiplication will grow, and eventually, the gradient will explode. The ReLU downing for this particular case by keeping the gradient value to one. In general, it can be seen that the derivative of the function has a value of one for the positive values and zero for the negative value. In the zero-value case, the function helps the network become sparse by nullifying some neurons connections, which is practically reducing the network size. The disadvantage of the weights nullifying is that some connections will be zeros, then many neurons will not function, and they will have no influence on the network. These neurons cannot contribute to network improvement.

#### 3.2.1.3 Leaky ReLU

The work of (Maas, et al., 2013) overcame the sparsity problem caused by the ReLU activation function by a variant of the same function called Leaky ReLU that is defined as:

$$f(x) = \begin{cases} ax & if\ x \leq 0 \\ x & if\ x > 0 \end{cases} \tag{3.9}$$

with the gradient of the from:

$$f'(x) = \begin{cases} a & if\ x \leq 0 \\ 1 & if\ x > 0 \end{cases} \tag{3.10}$$

where $a = 0.01$. The dying connection is addressed by allowing the function to return or "leak" a value of $a$ instead of zero, which is technically the gradient will be small but not zero. This property reduces the sparsity problem and, in the meantime, allows more robust optimisation since all the nodes will contribute to the network.

#### 3.2.1.4 Parametrised Activation Functions

These classes of activation function are suggested in (Zhang & Woodland, 2015). They suggested a generalized or trainable form of Sigmoid and ReLU function. For Sigmoid is:

$$f(x) = \eta . \frac{1}{1 + e^{-\gamma a + \sigma}} \tag{3.11}$$

which is the logistic function, and $\eta$, $\gamma$, and $\sigma$ are learnable parameters during the training. The function has the following properties as each parameter changes. If $\eta$ changes, then it will scale the dynamic range of the function itself. No constrain is applied to these parameters, so it can be even zero value, which means that the function will deactivate the unit. As it has been mention previously $\gamma$ will control the steepest of the function while $\sigma$ will horizontally bias the function, such that the value of the displacement above the x-axis will be $\sigma/\eta$.

The partial derivatives of the function of Equation (3.11) can be expressed as:

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 0 & if\ \eta = 0 \\ \gamma f(x)(1 - \eta^{-1} f(x)) & if\ \eta \neq 0 \end{cases} \tag{3.12}$$

$$\frac{\partial f(x)}{\partial \eta} = \begin{cases} (1 + e^{-\gamma x +} \sigma) & if\ \eta = 0 \\ 0 & if\ \ \eta \neq 0 \end{cases}$$
$$\frac{\partial f(x)}{\partial \gamma} = \begin{cases} 0 & if\ \eta = 0 \\ xf(x)(1 - \eta^{-1} f(x)) & if\ \ \eta \neq 0 \end{cases}$$
$$\frac{\partial f(x)}{\partial \sigma} = \begin{cases} 0 & if\ \eta = 0 \\ -f(x)(1 - \eta^{-1} f(x)) & if\ \ \eta \neq 0 \end{cases}$$

The parameterised ReLU consists of two portions of straight lines. One is equal to 0 when $x \leq 0$ and the other equal to $x$ when $x > 0$, each portion is parameterized with a learnable parameter as in Equation (3.13):

$$f(x) = \begin{cases} \beta . x & if\ x \leq\ 0 \\ \alpha . x & if\ x > 0 \end{cases} \tag{3.13}$$

Here, $\alpha$ and $\beta$ are learnable and similar to the parameterized sigmoid partial derivatives, and the parametrized ReLU partial derivative can be expressed as:

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \alpha & if\ \ \eta = 0 \\ \beta & if\ \eta \neq 0 \end{cases}$$
$$\frac{\partial f(x)}{\partial \eta} = \begin{cases} \alpha & if\ \eta = 0 \\ 0 & if\ \ \eta \neq 0 \end{cases} \tag{3.14}$$
$$\frac{\partial f(x)}{\partial \gamma} = \begin{cases} 0 & if\ \eta = 0 \\ \alpha & if\ \ \eta \neq 0 \end{cases}$$

Even though both $\alpha$ and $\beta$ are not constrained, they didn't create any issue like the gradient explosion. In the meantime, the gradient can be controlled by parametrizing the output of the function or using gradient clipping.

#### 3.2.1.5 Exponential Linear Units (ELUs)

The exponential linear unit with $\alpha < 0$ can be defined as (Clevert, et al., 2015):

$$f(x) = \begin{cases} x & if\ x > 0 \\ \alpha(e^x - 1) & if\ x \leq 0 \end{cases} \tag{3.15}$$

and it's derivative is shown as in Equation (3.16):

$$f'(x) = \begin{cases} 1 & if\ x > 0 \\ f(x) + \alpha & if\ x \leq 0 \end{cases} \tag{3.16}$$

The value of the parameter $\alpha$ controls the amount of saturation (or activation) of the function when the input value is negative. The effect of the ELUs is similar to the ReLU and LReLU. On the vanishing gradient case, the function's positive value is identical, and the gradient is one as in Equation (3.16); however, there are two differences. First, it has negative saturation at a small value of the input function, which means that the output will maintain its value around zero. It also means that the ELU can soften the interval covariance function's effect during the model training, as discussed in the batch normalization Section (3.4.6). Second, the derivative of the function is non-negative, meaning that the ELU activation function always pushes to the negative gradient direction (Ide & Kurita, 2017).

#### 3.2.1.6 Noisy Rectifier

Another attempt to remediate the issue in the ReLU is to add noise to the original activation function to be in the form of Equation (3.7):

$$f(x) = \max(0, z + x) \tag{3.17}$$

here $z$ is zero-mean noise sampled from logistic density function $p(z) = sigm(z)\big(1 - sigm(z)\big)$ where $sigm$ function is the logistic sigmoid function (Bengio, et al., 2013). Introducing this stochastic process in the activation function is to address the insensitive state in sparsely activated neurons. The insensitive state appears when the hidden unit output is zero after multiple backpropagation steps.

Two interesting properties can be explained for the arrangement of Equation (3.17). First, when $f(x, 0) > 0$, the response state of the function is normally active or sensitive, unless when $z$ is high enough to turn the activation function to zero (negative value such that $z + x < 0$). In the case that the system senses that $f(x)$ (or the hidden neuron $h$ ) at a particular iteration should be small, then the gradient will push the function to the non-activate or insensitive state. Second, when $f(x, 0) = 0$ that put the basic hidden unit in an inactive state. Now for a positive value of $z$ or $(z + x > 0)$, it pushes the argument of the rectifier above zero, then the gradient will flow and provide two states for that unit. If the unit has been active, which is not a good state that this unit is supposed to be in, the gradient will push the unit lower. On the contrary, if the active state is good for the unit to be in, it will push the weighted sum higher and increase its chance to be active again.

## 3.3 Regularization and Its Techniques

In any machine learning problem, the question that needs to be answered is how the trained model performs against a new set of data that has not been shown during the training phase. Reducing the test error is the mission of increasing the generalization of the model to a new sample by applying regularization techniques during the training phase of the model. In addition, regularization reduces overfitting but does not reduce the training error. Training a deep learning model implies extremely complicated domains like 2D images, 3D images, audio, and text. This means that finding the right model with the level of complexity that fits the input domain is not a simple task. Generally, a large model with well-regularized techniques results in an appropriate performance. Below are some of the methods of regularization that researchers use during the training phase of the model.

### 3.3.1 Parameter Norm Regularization

Deep Neural Network typically contains thousands or even millions of weights across its layers. Having a model with high capacity can easily overfit due to its tendency to memorise the training set. In this situation, the model losses its generalization and fail during the test phase. One of the techniques that address the tendency to memorise the training set is the weights regularization. In principle, weight regularization is applying constraints on the weights to prevent or make it hard for the model to optimise over the dataset. Generally, the concept can be performed by applying what so-called L1 or L2 regularization over a layer that can be defined as:

$$Regularization = \frac{1}{2}\beta\|\boldsymbol{W}\| \tag{3.18}$$

where $\beta$ is the regularization scale, and $\boldsymbol{W}$ is the weights vector. This function penalizes the weights (kernels and biases) and favors small weight values. Equation (3.18) can be added to the optimizer's total network function to calculate the weights updates and force small weight values. The equation mainly penalizes the large weight values since the norm is the square of the model weights. So, large values will contribute more to the loss function, and the simplicity of the L2 norm allows an easy way to differentiate and propagate the updates of the weights vector values. With the help of L2 regularization, the learning algorithm sees the input as having higher variance and shrink the weights accordingly. The effect of the L1 norm is entirely different in the learning algorithm. Compare to L2, L1 results in a sparser solution. Sparsity refers to an optimization solution that results in zero parameters, which is unlikely to happen with the L2 regularization.

### 3.3.2 Data Augmentation

One of the reliable techniques to increase the generalization of a model is to train it with more data as possible as it can be. In image or video datasets, the augmentation is performed by preprocessing geometric transformation, like random cropping, rotation, mirroring, and projection. Also, illumination and chromatic transformation like brightness, contrast, arbitrary color level transformation, and gamma transformation are part of data augmentation techniques. Combining all these operations provide a variety of fake data generated from the original dataset.

These transformations may fit some training operations like classifications but not other more restricted scenarios like the disparity. Even the rotation of stereo image samples with the same angle will break the epipolar constraint and spoil the rectification between the two images. Mirroring is not a recommended operation to perform in stereo because it may produce a negative disparity estimation in a network that estimating positive disparities. The cropping can be performed with careful attention. In other cases, the specific augmentation operation like rotation may not reflect real settings in the data during the training or the testing; therefore, it should be avoided. In cases like medical imaging, as in the case of body CT scan volume registration, the rotation may not be useful because it will not reflect the natural representation of data.

For many classification tasks, noise injection in the input data increases the model's robustness to noise. Another available method of noise injection is to slightly disturb weights within the hidden layers of the model with noise. Regularization with this technique reduces the effect of the small perturbations of weights to influence the output. Thus, the model becomes less sensitive to the small variation in the weights by seeking minima surrounded by a flat region.

### 3.3.3 Dropout

Another regularization technique that was first introduced in (Krizhevsky, et al., 2012) is the dropout that combats overfitting and allows better performance in many classification tasks. The dropout operation is performed at the FCNN part of the classification network by settings a portion of the activations between layers to zeros during the training process. Specifically, at each training sample batch, the zeroing is technically changing in a percentage of the connections randomly. Thus, part of the connections is participating in the prediction at each sample. This regularization technique prevents the network from depending on specific activations during the training since the activation may not appear for a particular sample twice.

Another explanation to why this technique provides regularization to a trained model is its inexpensive approximation of ensemble models to one model. Training a network that contains dropout settings is like training different networks by dropping a percentage of the connections between hidden layers. The equivalent of this operation is training a large number of small networks on the same task. The connection's dropping is performed by randomly multiplying the output of some of the neurons with a binary mask. The other explanation of dropout operation is that the dropout for a network consists of n units allows an exponential number of $2^n$ smaller networks share the same weights, which effectively improves the parent model regularization and increases its generalization.

One advantage of the dropout is that it is more effective than other regularization techniques, and it can be combined with other techniques for further improvement (Srivastava, et al., 2015). The second advantage is that it is cheap to compute, and it works with many types of networks, including CNN networks. However, the dropout performance reduced on low data size, and unsupervised learned features can perform better in this case than the dropout.

Applying the dropout technique enhances the network prediction capability for the unseen data by enforcing the network to learn redundant connections for its data samples. These redundant connections ensure that some of the information that may be missed in one activation may appear in another. During the training, the active dropout connections will be scaled to fit the dropped activation, while in the test, the dropout is not applied.

### 3.3.4 Pooling Layer

Pooling is a technique to reduce the spatial dimension of the feature without adding weights to the network. Thus, it reduces overfitting risk, and the computation as the size of the feature decreases spatially. Max pooling showed success in LeNet 5 network (LeCun, et al., 1998) and then used on other architecture like AlexNet (Krizhevsky, et al., 2012). Max pooling selects the highest value within the pooling window to generate smaller features with the max values from the previous features selected. The concept here is that features with high importance are with high activation and should be passed to the next layer while lower activation may be ignored. Zhou & Chellali, 2017 showed that max-pool and average pool might be used to improve the object recognition task's performance and robustness. Other pooling operations like L2 norm and weighted average are also available options.

The pooling layer allows the network to become less sensitive to a small translation in the input. In cases where the existence of the feature is more important than where the feature is, the pooling layer is the required operation. While this virtue is vital for image classification cases, it is not recommended in tasks like disparity and optical flow estimation. In disparity, the information depends

on these translations, and the location of the feature is the information needed to estimate. Thus, for these motion estimation features, the pooling layer is rarely incorporated.

### 3.3.5 Weight Sharing

In many training scenarios, the network may contain more than one input. Also, part of the data could appear on both inputs, and data may have the same information. It is reasonable to have equal weights that respond equally to the same input in such a case. As the L1 and L2 norm regularization affect the model's weights, weight sharing also regularizes the weights, but it has more advantages over the norm regularization method. The advantage the weight share possesses is the significant reduction of the model's memory footprint since only a subset of the parameters (the unique set) need to be stored in memory.

Weight sharing can be across models that perform closely related tasks in the multitask paradigm. This arrangement allows for better feature learning because each task influences the other (Zhang, et al., 2020).

### 3.3.6 Batch Normalization

The batch normalization technique is based on the assumption that each element of a layer in a neural network is normalized to zero mean and one variance with respect to its statistics within a mini-batch (Ioffe & Szegedy, 2015). Generally, the technique removes this covariate shift from the internal activations of the network. In other words, each activation is equipped with scaling and shifting parameters. To apply the batch normalization for each hidden unit $h_j$, both the mean $\mu_j$ and variance $\sigma_j^2$ over the mini-batch is calculated, then the hidden unit is normalized as in Equation below:

$$\hat{h}_j \leftarrow \gamma_j \frac{h_j - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}} + \beta_j \tag{3.19}$$

where $\gamma_j$ and $\beta_j$ are learned scaling and shifting parameters, respectively. Thus, the scaling and shifting parameters updates are also included during the backpropagation process. Generally, It has been shown that using batch normalization in deep learning improves the performance and speeds up the training process.

A new study (Santurkar, et al., 2018) showed that the batch normalization process has other impacts that improve a vanilla network. The study showed the vanilla network's loss function tends to have

large kinks, flat regions, and sharp minima in addition to its non-convex property. These characteristics make the gradient descents-based training algorithms unstable, thus, highly prone to explode or vanish. Furthermore, the training became more sensitive to learning rate and weight initialization. The batch normalization reparametrises the optimization process to a more smooth landscape. This effect reflects on the gradient to be reliable and predictive and helps to move with large steps without the risk of vanishing gradient due to flat surface or explode gradient due to sharp edges.

## 3.4 Optimization Algorithms

In machine learning or deep learning, the optimization algorithms are essential in minimizing the objective or loss function to an acceptable level. This objective indirectly helps a model to learn a mapping of the input to the output. To estimate how good the model in its mapping task, a performance metric needs to be employed during the training and testing phases to measure how well the model learned the data underlying distribution. The training metric (e.g., Least Mean Error) may be different from the testing metric (e.g., accuracy).

The optimization algorithm plays a vital role in estimating the required updates of the model weights and apply these updates such that the total loss function of the model is reduced. One of the fundamental optimization algorithms is gradient descent (GD), Equation (3.20):

$$w = w - \eta . \nabla_w J(w) \tag{3.20}$$

where $\eta$ is the learning rate. The GD's main issue is that it is prolonged because it is required to estimate all the derivatives of the mapping function concerning the model's weight for the entire dataset to calculate one update required for weight update. This is an expensive computation task, especially in a deep learning model containing millions of weight parameters (Ruder, 2016). Therefore, other optimizers are suggested like Stochastic Gradient

### 3.4.1 Stochastic Gradient Descent (SGD)

In the SGD algorithm, the weight update is performed at each training batch. Here, the loss function is differentiated with respect to the weights. The updates are distributed accordingly using chain rule but for only a small set of input (LeCunY, et al., 2015) without redundant computation of derivatives as in the case of GD. The input is defined by mini batches usually of $2^x$ in size where $x$ is usually integer to fit the computation device memory space. Therefore, the SGD requires only one derivative of the loss function with respect to the output, and then the backpropagation technique can update the weights accordingly.

$$w = w - \eta . \nabla_w J(w; x^{(i)}; y^{(i)}) \quad (3.21)$$

Here, $x^{(i)}; y^{(i)}$ are the input and the output instances. The disadvantage of the SGD is the suggested implication that the mini batch's expectation is equal to the expectation of the dataset in the limits. Using mini batches introduces variance in the update estimation and fluctuates the convergence, ensuring better local minima and improving the training speed (Johnson & Zhang, 2013). Therefore, it is crucial to reduce the learning rate $\eta$ at a proper time to reduce this fluctuation and use as high as possible batch size to reduce the variance fluctuation of the dataset. Furthermore, using momentum within the optimisation process can improve the convergence where the algorithm accelerates as it approaches the local minima.

### 3.4.2 Adaptive Moment Estimation (Adam)

In this optimization algorithm, each parameter in the model is computed with its adaptive learning rate. The algorithm stores the first and the second moments of the exponentially decaying average of past gradient and squared gradient (mean and uncentred variance), respectively. This optimizer is appropriate for non-stationary objectives and can cope with noisy and sparse gradients. The algorithm also requires little tuning (Kingma & Ba, 2014).

Loss function differentiation is the starting task of optimization algorithms. Thus, it is essential to have a loss function that should be smooth and differentiable function across the input range.

Despite the widespread popularity of Adam, recent research papers have noted that it may fail to converge to an optimal solution under specific settings. The article (Keskar & Socher, 2017) demonstrates that adaptive optimization techniques such as Adam generalize poorly compared to SGD. This has prompted some researchers to explore new strategies that may improve Adam (Bock, et al., 2018; Tato & Nkambou, 2018).

## 3.5 Deep Learning Approaches

There are many of the biologically inspired algorithms that are widely used in AI. Neural network developed in the fifties are the ancestors of the current wave of the technologies of the DL models. Over many decades efforts addressed different problems.

It has been mentioned previously that the input to the multi-layer neural network is in a vector form. Transforming images to be a vector to comply with this network means ignoring all the related information that the pixel forms with the neighbour pixels. This neighbour relation is an essential source of information that a model could utilize for training. Instead of destroying this related

information in images, researchers suggested a new structure that can accept the image as input (LeCun, et al., 1998). This new structure is the Convolutional Neural Network (CNN), which consists of convolutional layers alternating with pooling layers and ending with a multi-layer neural network that it called Fully Connected Neural Network (FCNN) in the context of the CNN. The CNN network accepts 2D or 3D images as input. Within this network paradigm, the network can be deep without a dramatic change in its weight. Furthermore, it allows weight sharing that increases the regularization by reducing the model degree of freedom.

In FCNN, the connection between the input and the output is full. Thus, for one neuron, the output equals the multiplication between the input vector and the weights related to that neuron. The convolutional layer changes the multiplication to a convolutional operation between two real value functions. The neural network libraries implemented the convolutional operation as a cross-correlation function defined as:

$$C(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n) \tag{3.22}$$

where $I$ is the input image, and $K$ is a kernel of weights.

In machine learning, the convolutional operation leverages three essential concepts. First, sparse connectivity, unlike the traditional neural network, the kernel size is very small compared to the input. Second, weight share, unlike the weight arrangement in the FCN where they are tied to only one input, the convolutional kernel shared by all the inputs and one set of the parameter will be learned for all the input locations. Third, the CNN kernel possesses equivariance to translation, which means that the output changes the same way as the input changes.

At the moments there ware many distinct approaches to implement DL models as below.

### 3.5.1 Autoencoder

One form of unsupervised learning with a neural network is the autoencoder. The concept of the autoencoder is a network that learns an efficient coding of its input. The coding of the input should be compressed or in reduced form compared to the input. The objective is to reconstruct the input image from its compressed representation. Thus, the output of the autoencoder is an approximated version of the input. The output of the autoencoder can be formulated as a probability problem. Therefore, the objective is to optimize the $p(\boldsymbol{x} = \hat{\boldsymbol{x}}|\tilde{\boldsymbol{x}})$, which is the probability that the model produces or gives the value of $\hat{\boldsymbol{x}}$ when the given value is $\tilde{\boldsymbol{x}}$ such that $\hat{\boldsymbol{x}} \cong \tilde{\boldsymbol{x}}$. Another interpretation of the autoencoder's

operation is to rebuild its input from a compressed representation developed in its hidden units (Goodfellow, et al., 2016).

Figure 3.1 shows a simple autoencoder that consists of encoder and decoder sides and a hidden middle layer with dimension $L$ that is smaller than $M$ to ensure the compression function. The encoder side of the equation can be expressed as:

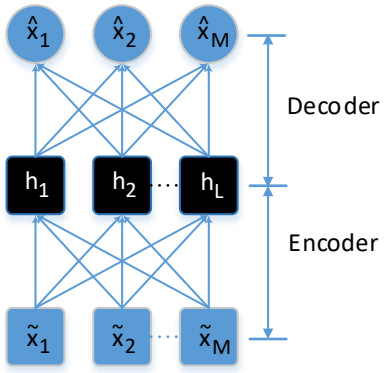



Figure 3.1: Simple Autoencoder Network.

$$Encoder = \boldsymbol{w}\tilde{\boldsymbol{x}} + \boldsymbol{b_1} \tag{3.23}$$

Similarly, the decoder side of the network can be expressed as:

$$Decoder = \boldsymbol{w^T h} + \boldsymbol{b_2} \tag{3.24}$$

Note that the parameter matrix $\boldsymbol{w}$ in Equation (3.24) is the transpose of its value in Equation (3.23). Also, the output of the network in this form will default to a linear network that can find the same subspace that the principal component analysis (PCA) algorithm can find. However, the ends of both the encoding side and the decoding side are rectified with activation functions to introduce nonlinearity to the network for better performance.

The function that describes the autoencoder is:

$$f(\tilde{\boldsymbol{x}}) = f(Decoder(Encoder(\tilde{\boldsymbol{x}}))) \tag{3.25}$$

and the probability equation can be written as:

$$p(\hat{\boldsymbol{x}}|\tilde{\boldsymbol{x}}) = p(x = \hat{\boldsymbol{x}}|\tilde{\boldsymbol{x}}; f(\tilde{\boldsymbol{x}})) \tag{3.26}$$

With Equation (3.26), an autoencoder model can be optimized using the negative log probability's objective function over the mini-batch dataset with stochastic gradient descent backpropagation.

### 3.5.2 CNN Architectural Evolution

Since the first network of LeNet-5, the structure of the CNN had a standard architecture that stack convolutional layers with activation followed optionally by pooling, or other layers. Then, the higher layers are usually fully connected, especially for classification tasks. Generally, referring to a model containing N layers is related to the number of convolutional layers within that model. Some of the models that impacted the network design concepts of the CNN include: AlexNet, VGG, Inception, RasNet, and SqueezeNet

These models are compared based on the object recognition task available by the dataset called ImageNet used in the ImageNet Large-Scale Visual Recognition Challenge: ILSVR-2010 to ILSVR-2015 competitions (Berg, et al., 2011). This dataset consists of 15 million labeled high-resolution images of 22000 categories. Usually, the comparison is based on a subset of this dataset with 1000 categories, making about 1.5 million images with 150000 images for the test. Two error rates are usually reported: the top-1 when the prediction does not represent the ground truth, and the top-5 when the ground truth is not within the first five predictions.

#### 3.5.2.1 AlexNet

AlexNet is the first successful model that was implemented to be trained with a Graphical Processing Unit (GPU) (Krizhevsky, et al., 2012). It also represents the initial start to a rapped progress and abundance of much deeper CNN network because the training time was a barrier while training with CPU became possible with GPU. The model used most of the techniques reported previously, like ReLU for activation, max-pooling, and dropout and weight decay functions for regularization.

The network architecture consists of 8 layers, five convolutional, and three fully connected layers. The last fully connected layer's output is fed to a 1000-way softmax function to produce the distribution over the subset of classes mentioned (1000 categories). The first convolutional layer kernel is of 11x11x3 with a stride of 4 for the input of 227x227x3 to obtain 55x55x96 features. A max-pooling layer of 2x2 and a stride of 2 is followed. Then a 5x5x96 kernels that output features of 27x27x256 then

another max-pool layer that results in a feature of 13x13x384 features size. For this feature, a kernel of 3x3 convolutional layer with a stride of 1 is applied twice, and the result is a feature size of 13x13x256. The resulting feature is changed to a vector and fed to 4096 neurons followed by two layers of fully connected layers of the same size of neurons, and then the output is applied to the 1000 softmax layer. All the convolutional layers and the fully connected layers are terminated with a ReLU activation. The dropout regularization technique is applied through the fully connected layers with 0.5 percent of drops.

#### 3.5.2.2 VGG

The VGG network consists of a network family that is based on 3x3 kernels across the layers of the network (Simonyan & Zisserman, 2014). These kernels are stride with 1 pixel and usually padded with 1 pixel to produce the same size of feature map at the output. Each layer is followed by a ReLU activation function and occasionally a max-pooling layer. The full connection portion is the same as the layers reported in AlexNet network. The family of the VGG consists of networks that contain 11, 13, 16, and 19 layers. The VGG-16 also includes two versions, one with kernels of 1x1 and the 3x3 kernels, and the other consists only of 3x3 kernels. They also arrange some consecutive layers to be stacked without pooling constructing a block of 2, 3, and even 4 layers. The utilization of a 3x3 kernel instead of the 11x11 kernel, as in the case of the AlexNet, is based on the receptive field concept. For 3x3 kernels stacked in the consecutive layers, their effective receptive field becomes 5x5 for the case of two successive layers, and 7x7 for the case of 3 consecutive layers, and so on.

With the arrangement of these stacked 3x3 layers, two advantages obtained:

- The function of a 3x3 block of layers is more discriminative than one layer with a 5x5 kernel.
- It implicates a regularization since a kernel of 7x7 can be decomposed to a 3x3 kernels.
- It decreases the number of parameters and increases the calculation of the module parameters.

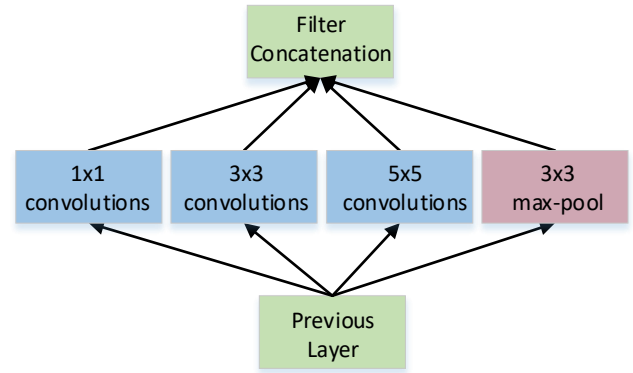
The network achieved a deeper architecture using this unified and straightforward kernel across the network.

#### 3.5.2.3 Inception

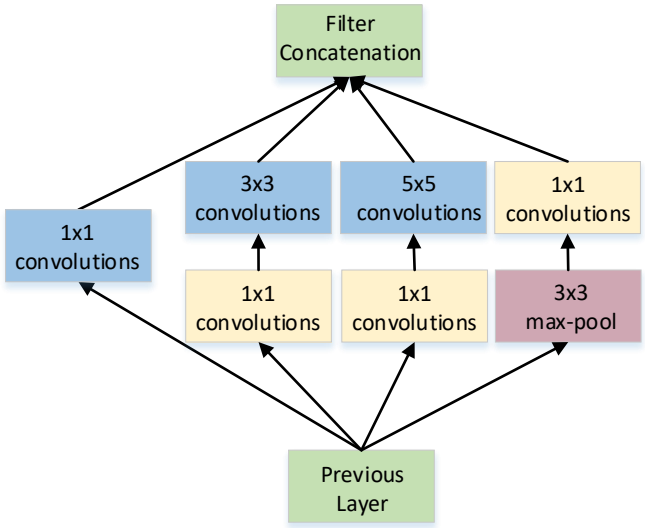
In this network, the target was to reduce the number of network parameters. A lower number of parameters means less requirement to computation power, lower overfitting chance, and less requirement to hug dataset for model regularization to train a large model. The inception network utilizes the concept of Network-in-Network to increase the representational power of the CNN by

introducing sparsity to the network and replace the fully connected network with sparse ones (Szegedy, et al., 2015).

The concept of sparsity is implemented by first clustering the input activation to statistically correlated units of features (Arora, et al., 2014). These clusters are statistically connected to the previous units and from the units to the next layers. These units are assumed to represent some regions in the previous activation and can be grouped into filter banks. Thus, it can be assumed that there are plenty of spatially small regions that can be covered by 1x1 kernels. Similarly, there could be a smaller number of spatially bigger regions that can be covered by 3x3 or 5x5 kernels. The suggestion can go on by stating that there will be bigger patches that can be grouped by bigger filter banks, but the designers of the Inception model stopped to 5x5 kernel size to avoid patch alignment issue. The output of all of these filter banks is concatenated and presented to the next layer to perform similar activation procedures to form a structure of the similar operation with what is referred to as the inception module. Furthermore, due to its success, a 3x3 kernel with one stride pooling layer is added to the inception module for regularization purposes, as shown in Figure 3.2 (a).




(a) Inception module, Naïve version




(b) Inception Module, dimensionality redaction

Figure 3.2: Inception Module (Szegedy, et al., 2015).

The module in Figure 3.2 (a) suffers from an issue. As the spatial feature's concentration decreasing towards the higher abstraction layers, the number of features needs to increase. This implication suggests that the kernels of the 3x3 and 5x5 will need to increase, which increases the computation burden. A 1x1 kernel was added to the inception module as in Figure 3.3(b) to overcome this issue. Here, to control the computation's bursting at the higher stages, a 1x1 kernel is added before the 3x3 and the 5x5 kernels to control the activation's depth using dimensionality reduction. In other words, the 1x1 kernel is compressing the feature map to a smaller number of channels before applying the computationally expensive kernels. This arrangement allows increasing the number of the units per each stage while controlling the parameters required to implement deeper models that could require more significant computation power otherwise.

In general, the Inception model consists of consecutive inception modules with occasional max-pooling layers of 3x3 kernels with a stride of 2. The model starts with typical convolutional layers, and the inception modules are added at higher stages. Before the softmax layer, an average pooling of 5x5

or 7x7 with different strides are used. This arrangement reduces the number of the module parameters substantially, which is reflected as less potential overfitting. To prevent the gradient from dying out, intermediate softmax layers have been introduced and trained with SGD algorithm. The network constructed with the inception modules coined GoogleNet, and it consists of 22 convolutional layers.

#### 3.5.2.4 ResNet

The main issue that appears with very deep models during training is the degradation problem. The case is that the model produces high training and testing error, while it is practically untrainable. The ResNet model addresses this problem by introducing the deep residual learning framework to overcome this degradation problem (He, et al., 2016). In its concept, the residual learning suggests that instead of multiple stack layers and hoping to fit the desired underlying mapping $H(x)$, it is easier to fit a residual mapping. That is, the stacked layers that fit the mapping $F(x)$ where:

$$\mathcal{F}(x) = H(x) - x \tag{3.27}$$

Thus, the original mapping $H(x)$ can be casted as:

$$H(x) = \mathcal{F}(x) + x \tag{3.28}$$

In an extreme case, the hypothesis is that the optimized residual mapping will be pushed to zero, then fitting the identity mapping obtained by a stack of nonlinear layers. Figure 3.3 shows the realization of residual mapping. Here, what is sometimes called a skip connection or a shortcut connection allows the identity mapping concept.

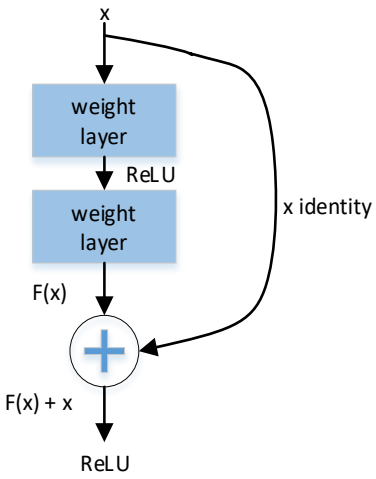



Figure 3.3: The residual learning building block.

The building block above is repeated every few layers in the residual module. The building block can be defined as:

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{3.29}$$

here, $x$ and $y$ are the input and the output of the building block, respectively. And the function $\mathcal{F}$ can be defined as:

$$\mathcal{F} = W_2\sigma(W_1 x) \tag{3.30}$$

where $\sigma$ is the ReLU activation function and $W_1$ and $W_2$ are the weights vectors of the first and the second layer within the residual block.

Thus, the deep model's optimization is turning to an identity mapping of a consecutive residual block. For instance, the identity mapping of Equation (3.29) can be achieved by reducing or nullifying the function $\mathcal{F}$ of Equation (3.30). The nullifying can be achieved during the training by weight regularization. With this arrangement, a network trained with 152 layers that consist of multiple residual blocks.

#### 3.5.2.5 Squeeze Nets

The ubiquity of the CNN in different appliances is increasing. The requirement to develop a lightweight model that can run with less demanding hardware like CPU power and memory requirement pushes to find less computationally expensive models (Iandola, et al., 2016). SqueezeNet is one of these solutions that achieved the same performance with much fewer parameters that the AlexNet

performed for top-1 and top-5 accuracy measurements. They designed SqueezeNet with three strategies in mind:

- Strategy 1. Replace the 3x3 filters with 1x1 filters, which reduces the number of parameters by a factor of 9.
- Strategy 2. Decrease the input channels to the 3x3 filters. This strategy is similar to the strategy used in the inception module, where the number of the parameters of the 3x3 or 5x5 kernels reduced by compressing the input channels using 1x1 filters. Thus, the number of parameters that are calculated as (3 x 3 x input channels x output channels) for a layer can be reduced by reducing the number of input channels.
- Strategy 3. The downsampling of the feature maps is delayed to higher layers in the network. An experiment by (He & Sun, 2015) showed that delaying the downsampling layers in different models implemented increased the models' classification accuracy.

Figure 3.4 shows the Fire module that was implemented to construct the network. The network also used identity mapping to construct a residual learning building block using the Fire module.

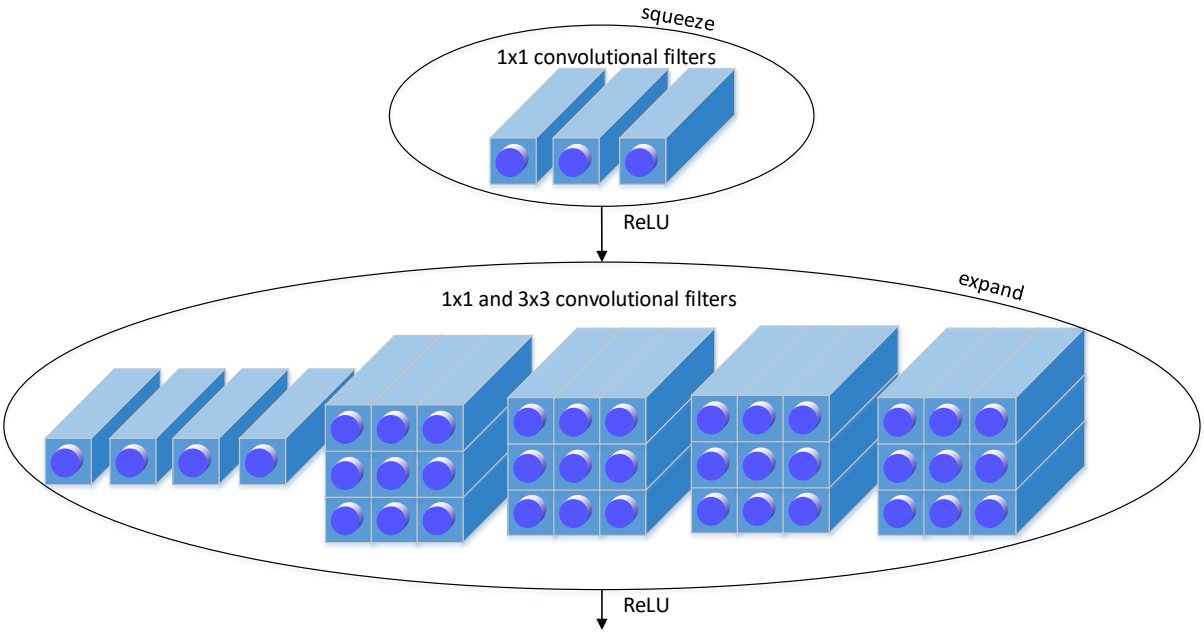



Figure 3.4: An example of a Fire module in the squeeze Net.

The other important detail of the network is that the network is not terminated with FCNN layers, and an average pooling is used before the softmax layer.

#### 3.5.2.6 Models Performance Comparison

The challenge of ImageNet has been considered the most effective approach to compare the performance of different network architectures in DL. This task represents a standard benchmark for deep learning. The comparison is based on the top-5 and top-1 predictions. Top-5 error is the percentage that the target is not appearing among the top probabilities of prediction. Similarly, the top-1 error is the percentage of the times where the target did not appear in the highest probability of the model prediction. Table *3.1* shows a comparison between the previously mentioned networks with their number of layers, number of parameters, and the top-5 error percentage.

| Network Name | Layers | No. of Param. | Top-5 Error (%) | Reference |
|---|---|---|---|---|
| AlexNet | 8 | 60M | 15.3 | (Krizhevsky, et al., 2012) |
| VGG Net | 19 | 138M | 7.3 | (Simonyan & Zisserman, 2014) |
| Inception v1 | 22 | 4M | 6.7 | (Szegedy, et al., 2015) |
| ResNet-152 | 152 | 69.2M | 3.6 | (He, et al., 2016) |
| Squeeze Net | 18 | ~120K | 15.3 | (Iandola, et al., 2016) |

Table 3.1: Well-known Network Architecture with their ImageNet competition performance.

While the previous object recognition method could not hit a 25% error, the first registered performance was with AlexNet with 15.3% improvement. It can be seen that as the model depth increases, the performance of the network improving. In addition, the performance increases as the kernel of 1x1 and 3x3 size are utilized in the deeper layers. With the human performance of 5.1% error (He, et al., 2015), it was the first time that a deep model outperformed a human recognition capability of 3.6% error with ResNet-152. On the other hand, the Inception model shows a better trade-off between the size of the model, its memory and processing footprint, and performance compared to the ResNet-152 number of parameters (Zagoruyko & Komodakis, 2016).

### 3.5.3 Restrictive Boltzmann Machine

Unlike deterministic neural networks like CNN, Restrictive Boltzmann Machine (RBM) is a stochastic neural network. These networks differ from feedforward networks by the existence of units that they are corresponding to random variables. They became popular before the outstanding results obtained by CNN. The RBM also has a deep implementation, and it appropriate to mention here. Figure *3.5* shows a simple representation of this network, a simple form of RBM structure that has similarity with the CNN in that the units within the layers are not connected in the level stage only between the units with different layers (Bengio, 2009).

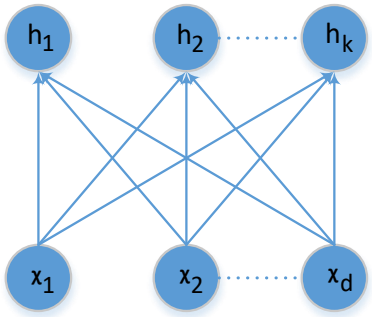



Figure 3.5: Simple Restrictive Boltzmann Machine (RBM).

To create RBM, the variables are partitioned to visible and hidden. A D-dimensional vector can define the visible variables, and the hidden ones can be defined as a K-dimensional vector. These two variables can be represented with a joint probability distribution of the from:

$$p(\boldsymbol{x}, \boldsymbol{h}; \theta) = \frac{1}{Z(\theta)} \exp\left(-E(\boldsymbol{x}, \boldsymbol{h}; \theta)\right) \tag{3.31}$$

where $E$ is an energy function that can be defined as:

$$E(\boldsymbol{x}, \boldsymbol{h}; \theta) = -\boldsymbol{x}^T \boldsymbol{w} \boldsymbol{h} - \boldsymbol{a}^T \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{h} \tag{3.32}$$

here, $\boldsymbol{w}$ is the visible-to-hidden variable interactions, and both $\boldsymbol{a}$ and $\boldsymbol{b}$ are the biases associated with the visible and the hidden variables.

The conditional probabilities $p(\boldsymbol{h}|\boldsymbol{x})$ and $p(\boldsymbol{x}|\boldsymbol{h})$ can be obtained from the underlying joint model as:

$$p(\boldsymbol{h}|\boldsymbol{x}) = \prod_{k=1}^{K} p(h_k|\boldsymbol{x}) = \prod_{k=1}^{K} p\left(h_k|b_k + \boldsymbol{w}_k^T \boldsymbol{x}\right) \tag{3.33}$$

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_{i=1}^{D} p(x_i|\boldsymbol{h}) = \prod_{i=1}^{D} p(x_i|a_i + \boldsymbol{w}_i \boldsymbol{h}) \tag{3.34}$$

Equations (3.33-34) are typically used to estimate the gradient of the loss with respect to the model parameters $\boldsymbol{w}$, $\boldsymbol{a}$, and $\boldsymbol{b}$ (Fischer & Igel, 2014).

### 3.5.4 Generative Adversarial Neural Network (GAN)

The theory of the GAN (Goodfellow, et al., 2016) is very appealing to researchers. It represents a method to change the unsupervised learning problem to supervised learning using a specific training scheme. The GAN's main idea is based on the game theory, where the generator tries to compete against an adversary agent. The generator network produces a sample $x$ by applying some input $z$ as in Equation (3.35)

$$x = g\left(z; \theta^{(g)}\right) \tag{3.35}$$

The discriminator network as the adversary competitor task is to discriminate between the sample $x$ obtained from the generator based on Equation (3.36), refer to as fake data, and samples drawn from the training dataset, the real data. The discriminator produces a probability value as in:

$$p = discr\left(x; \theta^{(d)}\right) \tag{3.36}$$

This probable value indicates whether $x$ been sampled from a real training dataset or the generator data distribution.

To formulate the game theory in this GAN setup, the function $v\left(\theta^{(g)}, \theta^{(d)}\right)$ is used to determine the payoff of the discriminator. The generator receives the $-v\left(\theta^{(g)}, \theta^{(d)}\right)$ as a payoff. An input sampled from the generator's data distribution becomes very close to the data's characteristics from the real training data. To achieve that, both of the parties must maximize their payoff, which could be concluded by the optimization representation of Equation (3.37) below:

$$g^* = arg \min_g \max_d v(g, d) \tag{3.37}$$

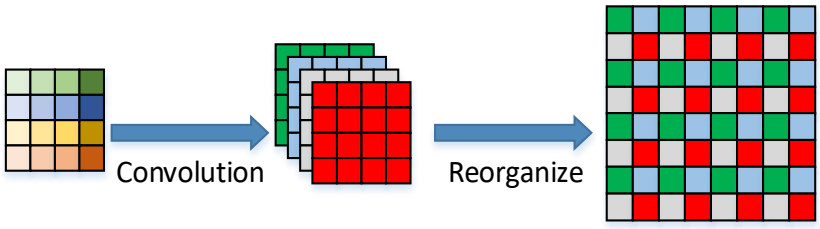
There are different implementations to Equation (3.37) that depend on the GAN variants that have lately developed.

## 3.6 Some Specialised Layers in DL models

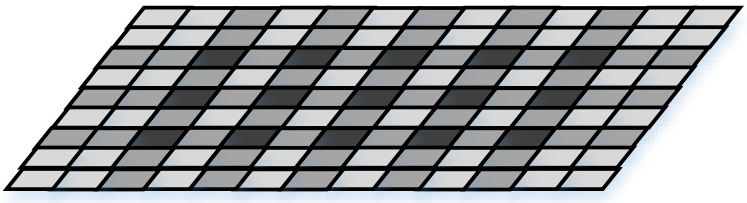
### 3.6.1 Reconstructing High-Resolution Images Form Low-Resolution Feature Set

The upsampling or the deconvolutional operations are usually the main elements in the decoding side of the fully convolutional neural network FCN. These layers allow upsampling the previous layers' features to increase their spatial dimensions, which are usually combined with reducing the features' depth. This operation is typically needed when the model has to generate an image or other computer vision functions from a low resolution to a higher resolution. An example of this technique is the subpixel convolution, where resulting convolution operations are reorganized to reconstruct the final image (Shi, et al., 2016) as shown in Figure 3.6a. Similarly, the result of the transposed convolution, which is essentially a reverse convolution, is shown in Figure 3.6b.

It has been shown that using the transpose-convolution or the deconvolutions and subpixel convolution suffer from what is referred to as the checkerboard artifact in the output image (Aitken, et al., 2017; Odena, et al., 2016). The checkerboard source on the output image is related to three reasons: deconvolution overlap, random initialization, and loss function.




(a) Subpixel convolution (Reorganizing or pixel shuffling)



(b) Transpose convolution or Deconvolution upscaled feature.

Figure 3.6: Checkerboard artifact. The effect of the checkerboard artifact on the output of some of the upscale convolution techniques.

The deconvolution or subpixel convolution overlap appears when the kernel size of the deconvolutional layer is not divisible by the stride step size, which causes the checkerboard effect. Even if the kernel is divisible, the random initialization causes the kernel to not activate in every pixel

of the higher resolution feature, which again causes the same effect. Finally, the gradient of the convolutional generates a checkerboard effect. Methods available to overcome this issue are reported in (Sugawara, et al., 2019)

A solution is available by upscale the feature with nearest-neighbor interpolation followed by a standard convolution with a stride of 1 step. This solution can remediate the first two causes since one stride will not allow the checkerboard effect, and the resulting pixels after the resizing will not be skipped. Therefore, researchers may perform the deconvolution with the upscale-convolutional combination in the image reconstruction step because of its superior results (Odena, et al., 2016).

### 3.6.2 Cross-Entropy Loss and Softmax Layer

The cross-entropy is defined as the distance between an original distribution and a model prediction or model belief output distribution. Typically, the original distribution can be represented by what is referred to as a one-hot vector. The one-hot vector is a vector of zero value or "false," and one or "True" to represent each hypothesis's probability that could be the correct prediction as in the classification task. The cross-entropy cost function for this binary representation can be defined as (Theodoridis, 2015):

$$J = -\sum_{n=1}^{N}\sum_{m=1}^{k_L}(y_{nk} ln\hat{y}_{nk} + (1 - y_{nk})\ln(1 - \hat{y}_{nk})) \tag{3.38}$$

where $n = 1,2,\dots,N$ and $m = 1,2,\dots.,k_L$. The minimum value of Equation (3.38) takes place when $y_{nk} = \hat{y}_{nk}$, which for binary target values is equal to zero. This cost function is the negative log-likelihood of the training samples.

For a vector of a target $\boldsymbol{y}_n \in \mathbb{R}^{k_L}$, with a single element in the vector equal to one corresponding to the target class concerning the input vector $\boldsymbol{x}_n$ and zero in the rest of the elements, the probability $P(\boldsymbol{y}_n)$ can be defined as:

$$P(\boldsymbol{y}_n) = \prod_{k=1}^{k_L}(\hat{y}_{nk})^{y_{nk}}(1 - \hat{y}_{nk})^{1-y_{nk}} \tag{3.39}$$

The cross-entropy function depends on the relative error, not on the absolute errors, which means that large and small error values are treated equally during the optimization step. Thus, the other expiration of the relative entropy cost can be defined as:

$$J = -\sum_{n=1}^{N}\sum_{m=1}^{k_L} y_{nk} ln \frac{\hat{y}_{nk}}{y_{nk}} \tag{3.40}$$

To guarantee the model's output is interpreted to probability, the sum of the output needs to be one. For this purpose, the softmax activation function is used to normalise the model output sum to one as the last layer in the model as in Equation (3.41):

$$\hat{y}_{nk} = \frac{\exp\left(z_{nk}^{L}\right)}{\sum_{m=1}^{k_L} \exp\left(z_{nm}^{L}\right)} \tag{3.41}$$

### 3.6.3 Correlation Layer

In many cases, when the network has multiple inputs, a matching feature could provide an important clue. The correlation process over two feature set captures correspondence over the optical flow or disparity space between the two sets. The first implementation of a correlation layer in deep learning appeared in (Dosovitskiy, et al., 2015). The main idea of the correlation layer is to implement the well-known matching technique in the field of optical flow and disparity estimation. The idea is to produce the feature sets for each input separately and at higher layers, the features correlated to create a joint feature. The correlated feature shows high responses only on the matching areas where both features are in the same place. Thus, features with close representation will provide high activation on the matching areas. The correlation layer performs a multiplicative patch comparison given the two feature set representations:

$U_1, U_2 : \mathbb{R}^2 \rightarrow \mathbb{R}^c$ then the correlation between two batches belong to two different features and centered on $x_1$ for $U_1$ and $x_2$ on $U_2$, and each of size $2k+1$ can be defined as:

$$c(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} \langle U_1(x_1 + o), U_2(x_2 + o) \rangle \tag{3.42}$$

Equation (3.42) is similar to the definition of the convolution, but it is different in that the patch is not convoluted (cross-correlated) with the kernel. Instead, it is cross correlated with another patch, and therefore, there are no weights involved in the operation, and the layer is not trainable.

To estimate $c(x_1, x_2)$, there are $K^2$ multiplication that needs to be performed and the correlation for an entire feature size of $w \times h$ requires $w^2 \times h^2$ of such computation or $K^2 \times w^2 \times h^2$ which is a very expensive operation in terms of memory and processing in both forward and backward operation. The displacement is mainly to limit the searching domain to be within the region of $(2d + 1)$ from $x_2$. Also, two strides performed, $s_1$ and $s_2$, where the first striding to quantize $x_1$ globally and the second is to quantize $x_2$ within the neighbourhood centered around $x_1$. The stride here is not the type of operation discussed in convolutional operation. The striding in this context of the correlation operation is performed during the estimation of the correlation value between the two batches, but the resulting feature map will be of the same size as the features $U_1$ and $U_2$.

The resulting correlation feature is four-dimensional, that is, for every combination of two patches, a correlation value is obtained with a total size of $batch - size \times w \times h \times (2d + 1)$.

### 3.6.4 Spatial Transformation Network (STN)

Spatial Transformation Network (STN) (Jaderberg, et al., 2015) is class axillary modules that can be attached to a deep learning module to perform various functions based on its position and the function of the STN. Essentially the STN consists of three components, Figure 3.7, as follows:

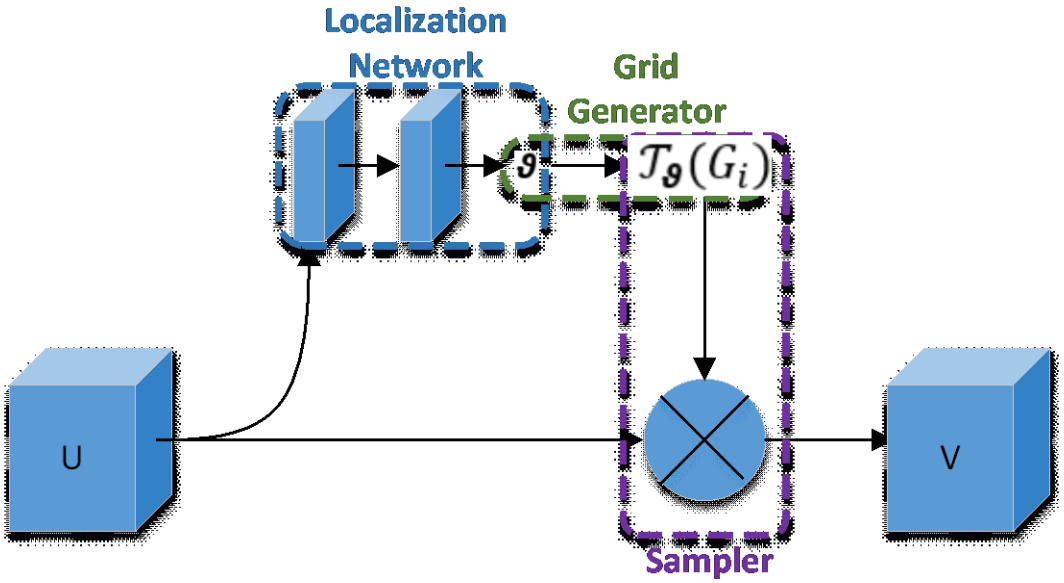



Figure 3.7: Spatial Transformation Network

- **Localization Network**: The localization network is a neural network. It can be constructed from multiple layers of FCNN or CNN, followed by FCNN as needed. The output of these networks varies based on the next components that the module is trying to learn. It could end with a six elements vector representing a transformation matrix of a rigid object or representing other transformations like Thin Plate Spline (TPS) or B-spline.

  The localization network takes the input feature map $U \in \mathbb{R}^{w \times h \times c}$ and output the latent vector $\vartheta$, where $\vartheta = f_{loc}(U)$ and size that changes based on the required transformation.

- **Parametrized generic sampling grid**: The generic sampling grid consists of locations (addresses) of each pixel from the input feature map that need to be warped or mapped based on a specific transformation $\mathcal{T}_\vartheta$. The dimension of the Grid G is like the feature dimension of size $U \in \mathbb{R}^{w \times h \times c}$. For the generic grid G, the $G_i = (x_i^t, y_i^t)$, target coordinates of the regular grid in the output feature map. The mapping is performed by applying the transformation function $\mathcal{T}_\vartheta$ to the generic gird G. Assuming the transformation function is an affine transformation, then:

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\vartheta(G_i) = A_\vartheta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \vartheta_{11} & \vartheta_{12} & \vartheta_{13} \\ \vartheta_{21} & \vartheta_{22} & \vartheta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{3.43}$$

  here, $(x_i^s, y_i^s)$ are the source coordinates in the feature $U$ that define the sample points and $A_{\boldsymbol{\theta}}$ is the output of the localization network that represents the affine transformation. Usually, the values of heights and widths are normalized. Thus both the spatial boundaries of the input and the output coordinates are between [-1,1] or $-1 \leq x_i^s, y_i^s, x_i^t, y_i^t \leq 1$. (Foley, et al., 1994).

- **Image Sampling**:

  The output feature map $V$ is obtained by spatially transforming the input feature map $U$ and reconstruct the new feature $V$ by interpolating the feature $U$ with the modified or parametrized grid $\mathcal{T}_\vartheta(G_i)$. Such that each pixel location $(x_i^s, y_i^s)$ represents the new position to the original pixel position in the $U$ feature set. The representation of this operation can be depicted as in Equation (3.44):

$$V_i^c = \sum_n^h \sum_m^w U_{nm}^c \, k(x_i^s - m; \Phi_x) k\left(y_i^s - n; \Phi_y\right) \forall i \in [1 \ldots . h'w'] \ \forall c \in [1 \ldots . c] \tag{3.44}$$

  where $k$ is the interpolation kernel and $\Phi_x$ and $\Phi_y$ are its related parameters. The interpolation can be bilinear, bicubic, or any other type. $U_{nm}^c$ is the specific value of the pixel

at the location (n,m,c) and $V_i^c$ is the corresponding pixel value at the pixel location $\left(x_i^t, y_i^t\right)$ at location c.

The STN can be used in different ways; that is, instead of using a separate network to deduce the transformation parameters $\vartheta$, the whole feature is used as $\vartheta$. In this case, that transformation will be performed directly on the generic grid but per pixel.

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = U_\vartheta + G_i = \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{3.45}$$

This implementation allows many applications that need pixel-level prediction and transformation to be implemented.

## 3.7 Sequence Modeling

Suppose the CNN is characterized by the convolutional operation represented by a kernel scanning the feature map without taking previous entries in consideration. The sequence model output is a function of the current input and the last output or hidden state. Since the last output is also a function of the previous input, the current output is also a function of the previous output and input.

In the RNN, the data sample consists of a sequence of words as in Natural Language Processing (NLP) or video sequence. A feed-forward network like CNN is not appropriate for this type of input. Networks like Time Delay Neural Network (TDNN) and RNN are addressing chronically related sequential data. In the meantime, RNN can process the image as a sequence if needed. For example, serializing an image (height x width) as a sequence of height-element input vectors with a timestep width can be considered an appropriate sequence to the RNN. As in convolutional operations, RNN sharing weights but in a different way. Here, the network consists of cells, and the output of each cell is a function of the previous output. Each member of the output is obtained based on the same update rule applied to the previous outputs.

### 3.7.1 Time Delay Neural Network

Time Delay Neural Network (TDNN) was first used for speech recognition problems (Waibel, et al., 1995). Few characteristics are required to implement a solution based on this type of network. First, the network should have multiple hidden layers with enough connection between them. This arrangement allows the network to learn nonlinear surfaces with the existence of nonlinear activation. Second, the network should have the ability to develop a relationship between events in time. Third, the developed features or abstraction should be time-invariant. Fourth, the number of network

parameters (weights) should be small enough compared to the training data to ensure proper encoding of the data and avoid overfitting. Fifth, the learning procedure should not require a rigid alignment to the target to facilitate network learning. Figure 3.8 shows a diagram of TDNN network.

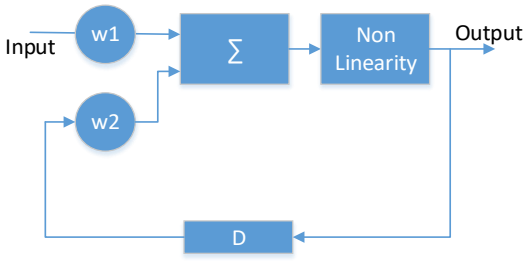



Figure 3.8: Time Delay Neural Network.

### 3.7.2 Recurrent Neural Network (RNN)

The development sequence models led to another structure called the Recurrent Neural Network, RNN. The RNN is based on a repetitive structure that is typically related to a chain of events. The sharing parameter of this structure can be revealed by unfolding the model.

The block diagrams of a generic RNN and its equivalent unfolded structure is shown in Figure *3.9* The unfolding model shows the equivalent parameter sharing in the time since the input passes through different values of the hidden states. Also, the model can be defined now with an acyclic computational graph.

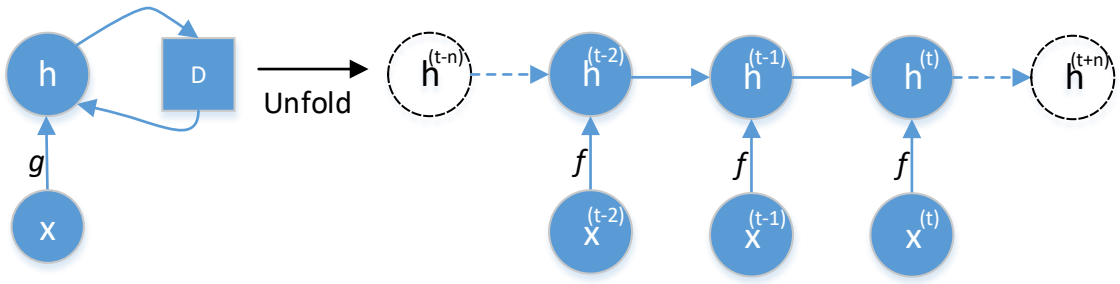



Figure 3.9: Recurrent Neural Network. Left is an RNN diagram with a delay unit (D). Right, it's an unfolded representation of the network.

Considering the diagram in Figure 3.9. The system equation can be described by:

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \boldsymbol{\theta}) \tag{3.46}$$

where $s^{(t)}$ is the current state of the network. In the same figure, $h$ is a hidden state of the model, and g is the transition function. The hidden state can be described, as in Equation (3.31):

$$h^{(t)} = g^{(t)}(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \ldots., x^2, x^1) \tag{3.47}$$

In the unfolded version of the network, $f$ is a transition function that its size depends on the sequence length and $h^{(t)}$ can be represented as:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \boldsymbol{\theta}) \tag{3.48}$$

In Equation (3.48), the current hidden state is calculated by the mapping action on the sequence $(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \ldots., x^2, x^1)$ as input. However, Equation (3.47) shows that the function $g^{(t)}$ can be factorized by applying the function $f$ repetitively. Equations (3.48) provide two important advantages:

1. The learning model is specified by the transition from one state to the next state, and it does not depend on the variable length.
2. The transition function $f$ and the share parameters are at every step of the training.

In this case, instead of learning separate models $g^{(t)}$ of Equation (3.47) at each time step, it is easier to learn single model $f$ for all the time steps and for all sequence length. With this strategy, the model $f$ can generalized to new sequences and with fewer model parameters.

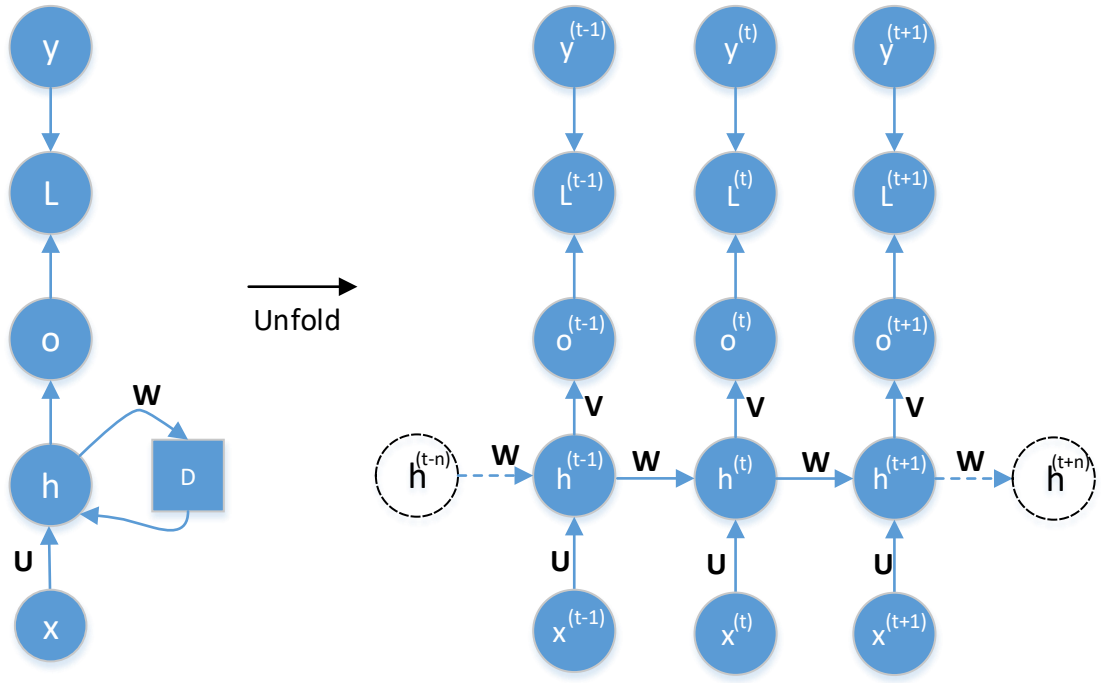



Figure 3.10: Recurrent Neural Network Equivalent Circuit. The model maps the input sequence 'x' to its corresponding output 'o'. The loss function L compare between the model output o and the ground truth y.

The RNN represented in Figure 3.10 is what is called the vanilla RNN model, and it shows the parameter arrangement with respect to the input between the state and the output. The input matrix $U$ has the same dimension as the instance $x$. Hidden-to-hidden is a square matrix, and V is a matrix in the dimension of the output. In the left of Figure 3.10 a folded model with the loss arrangement during the training. The right is the equivalent model, with each node, in this case, is associated with one particular time instance.

Let's consider that the activation function after the hidden state is the hyperbolic tangent, which is the case in most of the implementation of the RNN. Also, the output of "o" is a discrete unnormalized log probability of all the output's possible values. In this case, it is appropriate to use a softmax function as an activation function at the network's output. Having the input propagated in a forward direction and suggesting an initial value of $h^{(0)}$ and $t$ between $(1-\tau)$ the update can be expressed as:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \tag{3.49}$$

$$\boldsymbol{h}^{(t)} = \tanh\left(\boldsymbol{a}^{(t)}\right) \tag{3.50}$$

$$\boldsymbol{o}^{(t)} = c + \boldsymbol{V}\boldsymbol{h}^{(t)} \tag{3.51}$$

$$\hat{\boldsymbol{y}}^{(t)} = softmax\left(\boldsymbol{o}^{(t)}\right) \tag{3.52}$$

here, $\boldsymbol{b}$ and $\boldsymbol{c}$ are bias vectors, and the latent weight matrices $\boldsymbol{W}$, $\boldsymbol{U}$, and $\boldsymbol{V}$ are hidden-to-hidden, input-to-hidden, and hidden-to-output, respectively. Here, the input and the output lengths are equal, and the loss at a particular instance t is equal to:

$$Loss = -\sum_{t} \log p_{model}\left(y^{(t)} | \left\{\boldsymbol{x}^{(1)}, \ldots\ldots, \boldsymbol{x}^{(t)}\right\}\right) \tag{3.53}$$

where $p_{model}\left(y^{(t)} | \left\{\boldsymbol{x}^{(1)}, \ldots\ldots, \boldsymbol{x}^{(t)}\right\}\right)$ is obtained after reading the entry for $y^{(t)}$ from the model vector $\hat{\boldsymbol{y}}^{(t)}$. The main problem in this model is its expensive in gradient estimation and memory provision. The model is sequential; thus, each output depends on the previous one, which requires performing each step to go to the next step and, the weights states need to be saved to use them during the backpropagation. Furthermore, the lengthy process of backpropagation turns the process to suffer from vanishing or explosion of the gradient. This limitation has been resolved by developing the gated RNN. One of the successful solutions that proved performance in many applications is the Long-short Term Memory that follows.

### 3.7.3 Long Short-Term Memory (LSTM)

The main idea behind the gated RNN is to create a weighted path through time that change at each time step, which regulate the gradient and preventing it from vanishing or exploding. This network is the Long-Short Term Memory (LSTM), and it proved a success in speech recognition (Graves, et al., 2013) (Zhang, et al., 2017), machine translation (Wu, et al., 2016), image captioning (Gao, et al., 2017).

Figure 3.11 shows a typical LSTM cell.

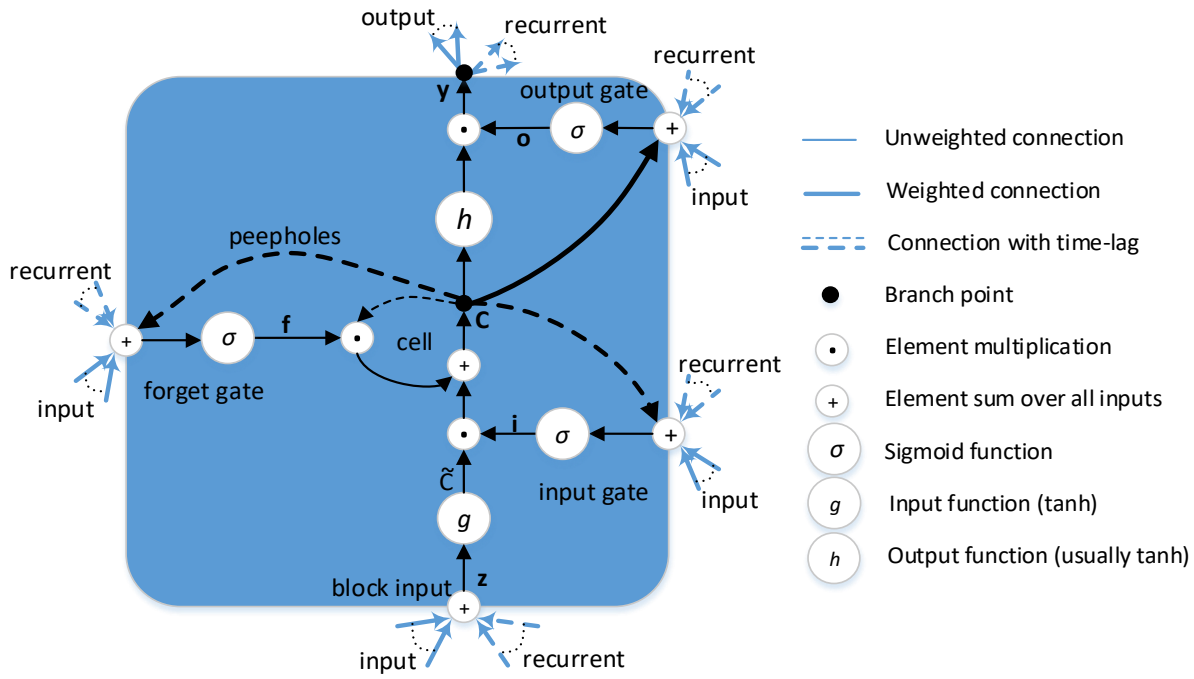



Figure 3.11: LSTM cell block diagram.

Considering the vanilla RNN, the LSTM cell replaces the hidden state in Figure 3.11. The cell in this figure represents a type of leaky unit that allows the network to accumulate information over a long period and use this accumulated data as a feature to contribute to the final output. In the meantime, the network can easily ignore or forget this accumulated information by reducing the weight that the feature is contributing by setting the weight value to zero. The loops within the cell facilitate the backpropagation and allow a long memory duration that can contribute to the decision without having a large memory footprint. It can be seen, though, that the cell has the same inputs and outputs that a typical RNN network has, but it has weighted gating units that control the information flow within the network. The forget gate can be defined as:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_i^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right) \tag{3.54}$$

Three gates are involved in each cell. The input gate, the forget gate, and the output gate. The input gate can be described as:

$$i^{(t)} = \sigma\left(\boldsymbol{W}^i\left[h^{(t-1)}, x^{(t)}\right] + b^i\right) \tag{3.55}$$

The input gate's essential function is to control whether the memory cell is updated. It depends on the weights multiplied by the concatenated version of the input and the previously hidden vector. The forget gate can be defined as:

$$f^{(t)} = \sigma\left(\boldsymbol{W}^{f}\left[h^{(t-1)}, x^{(t)}\right] + b^{f}\right) \tag{3.56}$$

where $f^{(t)}$ is the forget function with time step $t$. The current input vector $x^{(t)}$ controls the memory reset to zero if no previous states' contribution to the current prediction is required. Finally, the output gate is defined as:

$$o^{(t)} = \sigma\left(\boldsymbol{W}^{o}\left[h^{(t-1)}, x^{(t)}\right] + b^{o}\right) \tag{3.57}$$

The output gate controls whether the cell's current state should be propagated to the next state or suppressed. The mechanism of these gates is to control the flow of the information within the cell.

## 3.8 Summary

Deep learning improving and accelerating many research areas that were previously processed with other machine learning techniques. The maturity of the machine learning techniques that led to the development of the DL methodology and the new methods developed mainly for DL increases this algorithm's performance for many computer vision problems. Weight sharing in the convolutional layer and all the related regularization techniques are essential in the model training and the proper optimization for better performance. Spatial Transformer Network adds more functionality to the DL model to spatially manipulate the features such that the performance of the model can be improved. For timely related data, the convolutional model is not performing quite well. Recurrent Neural Network can process and extract temporal features but are limited in remembering previous events efficiently. In LSTM, most of the RNN problems had been addressed regarding remembering previous events and training efficiency. The GAN is a growing field in DL. The GAN will be utilized in this research for regularization in Chapter (6) with 3D convolution.

# Chapter 4 - BLINK DETECTION AS MOTION DETECTION PROBLEM

## 4.1 Introduction

In chapter 3, a selection of motion detection approaches was introduced, including a mixture of gaussian representations, optical flow, and spatiotemporal methods. In this chapter, motion detection is formulated using spatial and spatiotemporal representations. The first method is based only on spatial image features. In contrast, the second utilise these spatial image features to construct a temporal sequence of spatially coded features, or events, that are processed using a recurrent neural network. The proposed spatiotemporal method is similar to the video attention captioning network. In both cases, the problem of eyeblink detection is considered as a case study. In this chapter, the term 'eye status' refers to the eyelid's opening and closing action and is not related to an eye's health status.

In Section (4.2) a review of different techniques used for eyeblink detection. Sections (4.3) where the blink action analysis will be formulated that allow distinct solutions to the problems using DL methodologies. Sections (4.3-4) discuss the proposed spatial and spatiotemporal approaches for blink detection. Results are reported for the two proposed methods, and comparisons with other, previously reported literature and techniques are also included.

## 4.2 Related Work

An eyeblink is a temporal grouping of the eye status detection describing open-close-open action. Detection of eyeblink action has been of significant research interest with several important practical, real-life applications. For instance, a pilot must absorb information from various cockpit displays efficiently. It is therefore essential to tests the effectiveness of the cockpit display systems. The head & eye-tracking and eye status detection are instrumental in achieving that objective (Tan, et al., 2002). Other examples include face liveness detection (Szwoch & Pieniążek, 2012), helping disabled people to interact with computers (Grauman, et al., 2003), driver fatigue detection (Bergasa, et al., 2006), and assessment/monitoring of dry eye syndrome (Divjak & Bischof, 2009).

When detecting an eyeblink, the standard requirements are to apply the face and eye detection stages first. Generally, Viola-Jones (Viola & Jones, 2004) is the method of choice to recognize the subject's

face/eye areas. Also, the tracking of the face/eye can be performed to handle out-of-plane (yaw and pitch) (Saragih, et al., 2011) and/or in-plane (roll) head rotations (Tomasi & Kanade, 1991).

Most of the datasets incorporating the eye status (or eyeblink) exemplars contain the necessary annotation needed to extract the image areas representing eyes to perform algorithm development. These algorithms are mainly related to the motion detection techniques that address eyeblink detection or the eyelid motion in the video. The eye status (eyeblink) detection methods can be divided into two main categories, namely appearance and spatiotemporal based. These are briefly described in the following two subsections.

### 4.2.1 Appearance-based Methods

In this category, the researcher chooses features that reflect different pixel intensity profiles in the object image for their algorithms. Margins in the pixel intensities correspond directly to the brightness of light emitted from the object along with certain rays in space. Changes in these brightness profiles indicate a moving object.

The blinking rate is estimated by evaluating the eye status in the first stage, as in (Du, et al., 2008). Initially, the face and eye areas are detected. Subsequently, the area of the eye is cropped, resized to 10x30 pixels, and then binarized. The eye's blink evaluation for this method was based on the estimated ratio between the eye's length and its width. The authors reported 91.16% accuracy using this technique. Unfortunately, the dataset used for evaluation was not provided.

Similarly, (Lee, et al., 2010) described the eye status as open or closed using two features. First, they performed normalization of the illumination and binarization to have a meaningful estimation. Secondly, a Support Vector Machine (SVM) was trained to classify the features obtained from the first stage to estimate the blink action. The results showed a precision of 94.4% and a recall of 91.7% on the ZJU dataset (Pan, et al., 2007).

Danisman *et al*. (2010) also estimated the eye status to detect blinking. They utilized a specific eye geometry to extract features to train a Neural Network (NN), suggesting that when the eye is open, the pupil shows symmetry between the upper and the lower halves; when the eye is closed, this symmetry changes. Therefore, the difference between the upper and the lower halves is used to create the necessary features to train the NN. The algorithm achieved 90.7% precision and 71.4% recall on the ZJU dataset.

Fazli & Esfehani, 2012 suggested that if the face image is divided into five horizontal regions, the eye location will appear on the face's third and fourth subdivisions. They detected white and black pixels

in the mentioned area and related them to the eyelash and the iris. After locating the eye's relevant areas, the image is converted to a grayscale, and a suitable threshold is applied to estimate the number of white pixels representing the sclera. The authors reported success rates between 94.93% and 100% on four captured videos with 720x1280 pixel resolution.

In work done by (Dinh, et al., 2012), researchers employed their appearance-based model. This method assumes that the iris' intensity vertical projection values (IVP) are lower than the surrounding regions. The IVP contains two local minima during opened eye, but the function's shape will be monotonic when it is close. The changes in the profile of this function employed for blink detection.

Other appearance-based approach has been achieved by measuring the distance between two ellipses models that fit the eye pupils (Fitzgibbon & Fisher, 1996). To fit the model, they implemented a state machine technique.

In (Marcos-Ramiro, et al., 2014), the classification of each pixel using offset sampling technique is implemented to obtain four classes as follows: iris and pupil, sclera (white part), eyelids, and background (i.e., non-eye regions). Then they trained a random forest to perform the classification to detect the blink.

Another appearance-based model for blink detection is based on the Local Binary Patterns (LBP) (Malik & Smolka, 2014). Here, for each pixel, a binary pattern is created, as shown in Figure 4.1.

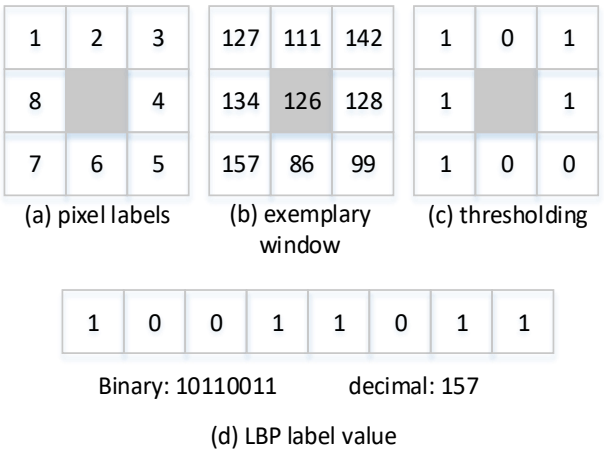



Figure 4.1: Local Binary Pattern label example.

Generally, the LBP can be defined as:

$$LBP_{Q,R}(x) = \sum_{q=1}^{Q} s(I_q, I_x).2^{q-1} \tag{4.1}$$

here, $Q$ is the sampling point of a circle of radius $R$ with respect to the centre $x$. $I_q$ is the pixel intensity surrounding the central pixel $x$ with intensity $I_x$ and and $s$ is defined as:

$$s(\xi_1, \xi_2) = \begin{cases} 1, & if \quad \xi_1 - \xi_2 \geq \delta \\ 0 & otherwise \end{cases} \tag{4.2}$$

Following the LBP estimation, the texture descriptor can be built by constructing a histogram of the LBP labels. To estimate the blink, the distance between the histogram of the opened eye and the closed eye is used to detect the blink action. They measured the histogram vectors' distance with Kullback-Leibler (KL) Divergence (Dahlhaus, 1996) and smoothed the resulting signal with the Savitzky-Golay filter (Press & Teukolsky, 1990). Subsequently, they identified the blinks by estimating the local peaks by performing Grubb's test (Tietjen & Moore, 1972). The method achieved 99.2% detection accuracy on the ZJU database. The main disadvantage of this method is that it works offline.

Another appearance-based study has been reported in (Borza, et al., 2018), where two models were incorporated to estimate a blink's occurrence. The first model depends on the estimation of the ratio between the eye height to its width. If the eye open, then the height is high. They utilized the dlib face analysis (Baltrušaitis, et al., 2016) toolkit for the eyes' landmarks to estimate the ratio. The second model that they used is based on CNN and trained for opened and closed eye classes (two classes). The model consists of four layers, with the first layer as a conventional neural network, while the rest of the layers contains a depth-wise convolutional neural network. All the layers contain kernels with a 3x3 pixel size, and the model is terminated with a softmax layer. The dataset used was Closed Eyes In The Wild (CEW) (Song, et al., 2014) with an image size of 24x24 pixels at the model input. They merge the results of the two models with a ratio of 0.75 weight to the CNN output. They reported a fluctuation of the CNN model's output to images with a degraded resolution that introduces false positives. This degradation problem encouraged to seek a sequential based solution (Anas & Matuszewski, 2018).

### 4.2.2 Spatiotemporal Based Methods

In this approach, the blink detection is based on the motion analysis of the eyelid. Following this approach (Drutarovsky & Fogelton, 2014) extracted features and passed them to the Kanade-Lucas-Tomasi (KLT) tracking algorithm (Suhr, 2009). The detection method exploits the motion vector components in the direction of the intensity gradients between consecutive frames. To perform the tracking, the eye image is separated into 3x3 cells of regions, and a flock of nine KLT trackers is placed over a regular grid with a spacing of 1/15 of the region dimensions. Then the motion related to the position of these trackers is calculated and averaged. The average variance computed for the regions

where the eyelid is located used to detect an eyeblink. The variance of the motion vectors that are related to the eyelid motion had been compared to an empirically selected threshold value to evaluate the eyeblink action. The implementation included a state machine to perform the status detection part of the process. The method achieved a precision of 91% and a recall of 73.1% on the ZJU dataset. The authors also developed a dataset called Eyeblink8 (Fogelton & Benesova, 2018). On this dataset, they showed 79% precision and 85.27% recall.

Fogelton & Benesova, 2016 improved blink detection by estimating the motion vector for each pixel's within the eye image region. To normalize the motion vector, an estimation of the intraocular distance was performed by estimating the eye corners' position. To calculate this distance between these corners to find the intraocular distance. The normalization helped to minimize the effects of other movements, like a head motion on the eyeblink estimation. As in (Drutarovsky & Fogelton, 2014), a similar state machine was adapted to estimate the eyeblinking. They reported results showed 100% and 98.08% for precision and recall on the ZJU dataset, 94.7% and 99% on the Eyeblink8 dataset, 95% 93.44% for the TalkingFace dataset. The authors also developed their dataset coined “Researcher‘s Night” and reported 92.42% and 81.48% for precision and recall, respectively.

In another study by (Fogelton & Benesova, 2018), an RNN had been trained to detect the blink action. They tried different features to train the RNN and obtain the best results. First, they developed features based on the work they reported in (Fogelton & Benesova, 2016) and developed a 361 feature vectors. Second, a weighted gradient descriptor has been developed by subtracting the eye region of the consecutive frames. The resulting frames are normalized to be between [-1,1]. Third, another type of feature that takes the aspect eye ratio is developed similar to the work performed (Borza, et al., 2018). Fourth, Histogram of Oriented Gradient (HOG) features generally performed the best as reported by the authors.

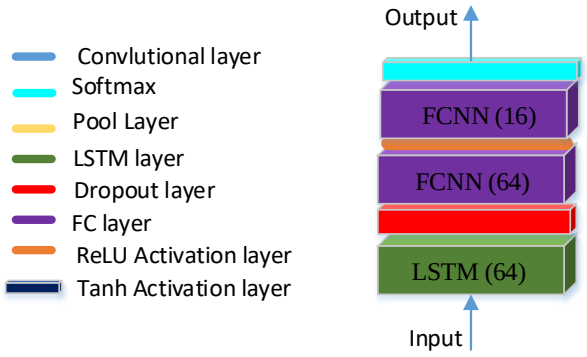



Figure 4.2: The LSTM Network reported in *(Fogelton & Benesova, 2016)*

The input consists of ten consecutive frames of the Researcher's Night dataset (Fogelton & Benesova, 2016). They chose ten frames as input for two reasons. First, more instances (frames) at the input means more temporal features that can be extracted. Second, they found that 98% of the Researcher Night dataset's blinks are within ten frame space.

## 4.3 Eyeblink Estimation

It has been mentioned in Chapter 2 that motion can be detected as an image intensity perturbation within an area related to a specific object across two or more frames. In this sense, eyeblink can be detected as an image intensity variation within an area of the detected eye, e.g., caused by iris appearing and disappearing as a result of eyelid motion obstruction visibility of iris and pupil (Rahman, et al., 2015 ; Du, et al., 2008). The eyeblink duration can vary from one person to another and for different eyeblink instances of the same person. There is no upper limit on the duration of an eyeblink clearly defined. Table 4.1 provides basic eyeblink duration statistics computed for three popular datasets. Furthermore, the sequence of eye events may not be precisely described, as in the case of incomplete blink when the closed state does not appear in the blink sequence, or extended blink when the blink action takes longer to execute (Portello, et al., 2013).

(Fogelton & Benesova, 2016) stated that an endogenous or a normal blink could last for 400ms. It means that at 30 fps, the blink action can appear in (30 frames/s * 0.4s) (i.e., about 12 frames). Others classify eyeblink as a short or long blink (Królak & Strumiłło, 2012). The short blink is considered to last up to 200ms and is called a spontaneous blink, while the long one lasts more than 200ms. Others suggested categorization of the blink into three groups, "awake" with blink duration shorter than 400ms, "drowsy" with blink duration between 400ms and 800ms, and sleepy for blinks that last longer than 800ms (Soukupova & Cech, 2016). Although the choice of the number of frames in which the blink action appears is a key factor in the detection capability and eventually in the proposed solution's performance, in many implementations, selecting the number of frames representing eyeblink is subjective.

In this research, the blink period is based on the available ground truth provided by different research datasets listed in Table (4.1), showing the average blink duration in ms.

| Reference | DataSet | Average Blink Duration (ms) $\pm$ Standard Deviation (SD) |
|---|---|---|
| (Divjak & Bischof, 2009) | ZJU | $391 \pm 135$ |
| (Divjak & Bischof, 2009) | Talking Face | $390 \pm 254$ |
| (Soukupova & Cech, 2016) | EyeBlink8 | $339 \pm 157$ |

Table 4.1: The ground truth of two well-known blink benchmarks.

Eyeblink detection is usually addressed by one of the two approaches. It is achieved either by identifying the eye state in each frame, searching for the sequence of interest, or processing multiple frames analysing the eyelids' motion. The action can be defined as continuous changes in the state of the eye between opened, partially opened, and closed and back to open or to partially opened state. A complete blink occurs when these states' sequences are completed within a typical timeframe between 100 and 400 milliseconds. In some cases, the sequence may not be complete as described above, like the case of the incomplete blink and the extended blink cases, when either the closed state does not appear or when the blink action takes a long time to be executed.

### 4.3.1 Eyeblink Detection Requirements

Generally, from the perspective of practical system applicability, an eyeblink detection system should (Sun, et al., 2013; Rezaei & Klette, 2012; Mohammed, 2014).

- Cope with changes in the illumination conditions,
- be invariant, within limits, to head pose changes,
- be resilient to motion blur
- operate in real-time,
- can perform in different frame rates,
- be independent of the camera used.
- The experimental results should reflect a real operational environment like imperfect face/eye detection.

In many cases, the evaluation of existing methods can face real challenges in real-world implementation. For instance, in the eyeblink detection field, using annotated eye image data to test a method is a common practice. These annotations are usually perfect and provide good quality eye images, which can explain the excellent validation results reported in some papers. However, when the face or eye detectors are used in a live scenario, the following could happen:

- The eye is not fully visible in the detected area.
- The eye is not detected, e.g., when it is closed (eyelid down)
- Detection of variable eye size in consecutive frames.

Illumination, motion, and other factors affect the overall detector performance, which eventually affects the system performance. For these reasons, most of the eyeblink solutions would not perform well in a real environment. The issue is that the cropped eye image from the face image may be obtained with large localization error and all other problems mentioned, potentially hindering these algorithms performance (Rahman, et al., 2015 ; Du, et al., 2008).

### 4.3.2 Eyeblink Benchmark Datasets

One of the datasets that are frequently used as a benchmark for the assessment of eyeblink detection algorithms is ZJU (Pan et al., 2007). Other eyeblink benchmarking datasets that appeared later, e.g., the Eyeblink8, or have been repurposed, for eyeblink benchmarking, such as the TalkingFace dataset. These datasets met most of the requirements listed above to provide insight into the solution's validity for real application. Each dataset has different characteristics that can affect performance in different ways. Furthermore, the subjects within each dataset show different blinking characteristics. Besides, the same subject may change his or her blink pattern within the same video, which is uncommon in a normal setup. From a practical point of view, such variability indicates the solution's capability to manage different blink patterns. For instance, a subject can perform a complete blink cycle, with both eyes closed and then opened. In other cases, one eye could be totally closed while the other is partially open. When both eyes are partially open (or partially closed), this case is defined as an incomplete blink cycle. This “unsynchronized” behavior makes blink detection difficult, such that merely using eye state to detect eyeblink is non-sufficient to deal with these variations.

#### 4.3.2.1 TalkingFace dataset

The TalkingFace dataset (Group, n.d.) consists of 5,000 frames for one subject. The images' resolution is 720x576, captured from a 25-fps camera, making the total video length 200sec. The dataset does not include a blink annotation, so the necessary annotation has been performed by observing each frame and compiling a text file containing the annotations made during the test. The total number of blinks observed during the annotation was 63. The subject in this video moves his head and makes different facial expressions.

#### 4.3.2.2 ZJU dataset

The ZJU dataset consists of videos for 20 subjects, each with four frontal-view recordings with the camera in different positions (Pan et al., 2007). Also, for each subject, there are two videos with glasses and two without. Thus, the total number of videos in this dataset is 80. The video of each subject is slightly different in length, with an average of around 15sec. A 30 fps camera took the videos with a resolution of 320 ×240. Due to this low resolution, the face and the eye detector find it difficult to work correctly. The subjects are steady and do not move. The only movement is the eyelids, with some background movement due to other subjects moving. Two hundred fifty-five blinks were identified in these 80 videos.

#### 4.3.2.3 Eyeblink8 dataset

The Eyeblink8 dataset was specifically built for blink validation. It consists of 8 videos captured on a 30-fps camera with a 640x480 frame size. Similar to TalkingFace dataset, the subjects in Eyeblink8 perform different facial expressions and some other activities like drinking. They are four subjects, one man and three women, each with two videos. Drutarovsky & Fogelton, 2014 reported 355 eyeblinks, while Fogelton *et al*. (2016), after manual annotation, reported 390. This dataset consists of 8 videos in the following sequence: videos 1,2,3,4,8,9,10,11. The total number of ground truth evaluated for videos was 408 blinks, as listed in Table 4.2. As it shows, each researcher uses different characteristics to calculate the number of blinks. While some counts subtle change in the eyelid as a blink, others want to see a complete blink cycle.

| Video No. | Video 1 | Video 2 | Video 3 | Video 4 | Video 8 | Video 9 | Video 10 | Video 11 |
|---|---|---|---|---|---|---|---|---|
| No. of Blinks | 35 | 86 | 51 | 31 | 31 | 42 | 66 | 66 |

Table 4.2: Videos chosen for testing the model blink detection.

## 4.4 Eyeblink Detection with Separate Spatial then Temporal Features

The main issue in the eye blink predictions in a video is the sparse occurrence of the blink event in a long video sequence. For instance, in a video of ten seconds, there may be 300 frames (for a 30 fps rate) that may contain only one blink. Extracting spatiotemporal features for this event in this large domain is challenging. It can be much easier to capture the eye's status at each frame, effectively reducing the dimensionality of the input domain from 300 frames to detect a blink to one frame to detect the eye status spatially. This novel approach represents an effective way to reduce the dimensionality of the input data while keeping the suggested models for this task in a reasonable size.

The first proposed method detects eyeblink using the CNN-RNN processing pipeline (Anas & Matuszewski, 2018). In this approach, it is to suggest a case where a CNN part is already trained to address a specific classification task of object states, like detecting the eye state, “Opened”, “Closed,” and “Partially opened.” Applying a sequence of images of the object states allows events detection that the object experiences, like blink event. The input to the CNN model can be spatial information like image sequence. Each input element in the sequence (image) will be mapped to an output (class).

The mapped input sequence generated by the classification model can be utilized as a new dataset to a sequence-based training model. The sequence-based training model can receive the output of the classifier model and extract temporal features. In this case, CNN, as a classifier, will extract spatial information about the sequence. Then, a CNN's latent representation will be organized chronologically

as sequences to extract the temporal information using RNN. Thus, a pipeline model will perform spatial than temporal processing in two separate networks.

The classification CNN is first trained to detect three eye states (Anas, et al., 2017). This stage represents a spatial inference of the spatiotemporal task. The AlexNet was selected as the classification CNN and trained on three classes, “Opened,” “Partially opened,” and “Closed.” Each class is defined by a percentage of iris being visible, as is shown in Table 4.3.

| Eye Status | Iris Visibility % |
|---|---|
| opened | ~100-50% |
| partially opened | ~50-5% |
| closed | ~5-0% |

Table 4.3: Definition of eye status based on iris visibility.

Figure *4.3* shows features detected by the AlexNet CNN corresponding to different eye states. It can be seen that these features are mainly representing eyelid’s edges. This suggests that the network learned to follow eyelids shape and ignore skin colour as skin colour does not provide important information about the eyes state. Although neuron activation may appear at different network locations, they are mainly related to eyelid edges' appearance (Anas, et al., 2017).

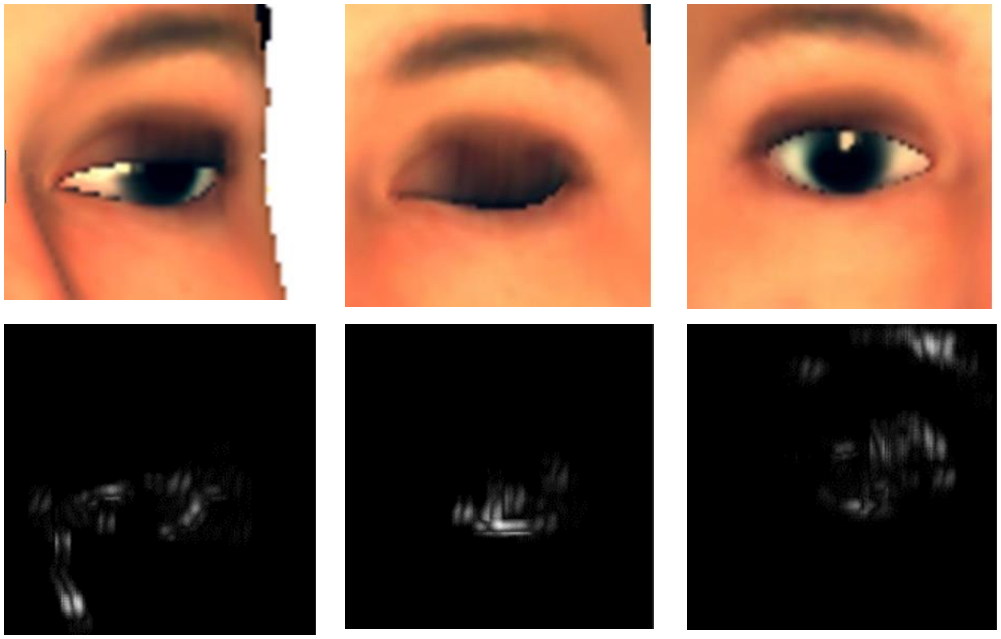


Figure 4.3: CNN eye state Activation. Image features reconstructed from the single strongest activation selected from the last pooling layer (bottom row) for exemplar synthetic images representing each eye state class (top row) (Anas, et al., 2017).

### 4.4.1 CNN-RNN Architecture

The eyeblink action can be modelled as a sequence of eye states. For instance, opened, partially opened, closed, partially opened, and opened set of states. Such set can represent a “true” eyeblink events. It is then the matter of detecting such sequences in a video to indicate an eyeblink action. The CNN's output, trained for eye state detection, can be any of these three classes. These classes' sequences can subsequently train a recurrent neural network (RNN) to identify the temporal relation between the eye states detected in each frame.

To illustrate, the output of the classification CNN has been analysed. In Figure 4.4, the outputs from the last layer of the CNN, the softmax layer, for images representing the same class of “Partially opened” are shown. The probability of state occurrence for a given image is more informative compared to the state information. For instance, in the first case (image on the left), the eye is almost opened, while in the second case (image on the right), the eye is almost closed. In this case, the probability contains information about the “Opened” and “Closed” states that could be used to differentiate between these two images, representing the same discrete “Partially opened” eye status.

closed 3.97%

opened 17.78%

partially opened 78.25%

closed 7.51%

opened 0.82%

partially opened 91.67%

Figure 4.4: CNN classification predictions for eye-opening, for the two synthetic images shown, the eye state is given as ‘partially opened.’ However, as the eyelid is closing, the ‘close’ state probability (confidence level) increases. Similarly, when the eyelid is opening, the ‘open’ state probability is increasing, i.e., getting more significant in the prediction state. The use of the state probabilities provides more meaningful information about each state's importance when combined with results from the neighboring frames.

The output of the softmax layer (CNN last layer) is commonly summed to one as it represents the probability of occurrence of each class. As the score value of one class increases, the contribution of the other classes on the distribution will decreases in a mixture of gaussian distribution. If the model is not sure about any class, the gaussian distribution will be almost with the same distribution across the three classes. The probability of the occurrence of the eye state can provide the latent information as it shows the weight of one state with respect to the other states. This relationship can be better

represented as a gradual transition in the eye state sequence. As a result, the CNN model output probabilities are used to develop a training set for the RNN discussed next. The range of the CNN output is between [0, 1] since the final CNN layer is softmax. The CNN outputs three numbers for each frame as classifier, corresponding to three possible eye states. Using both the left and the right eye, six numbers are assigned to each video frame.

Based on observations reported in Table 4.1 and considering the duration of the shortest blink; seven frames window has been adopted as a temporal length of data sample for the RNN. This arrangement gives latent vectors of 42 elements (6 x 7) being fed to the RNN. The RNN network is trained on about 1400 sequences generated from randomly pooled images from the three classes.

Figure 4.45 shows the CNN-RNN processing pipeline that includes the classification CNN followed by RNN for processing as a single data unit. RNN architecture is shown in Figure 4.5. The RNN processes 42 element vector for each new frame available at the CNN input.

### 4.4.2 Classification CNN Model Variants

As mentioned before, AlexNet CNN was used as the classification network in the proposed CNN-RNN architecture (Anas, et al., 2017). However, another classification network was also implemented to examine if there is any significant bias toward specific classification network architecture as well as to illuminate the resources required for a GPU based implementation, Figure 4.7. The newly designed CNN called ConcisedNet. It consists of a smaller number of layers and requires less computational resources, and is trained on the same way that AlexNet network trained with (Anas, et al., 2017).

### 4.4.3 RNN Training and Operation

The training of the RNN is performed over the available EyeBlink8 dataset. In particular, videos number (3,8,11) has been used to develop the numerical sequences to train the RNN network. Image sequences extracted from the videos mentioned above and first augmented to generate around 1400 image sequences using cropping and color modification and then passed through the ConcisedNet network to generate their corresponding latent sequences appropriate to the RNN training. The ground truth of this dataset is also organized to fit the blink event sequence. The reason for choosing videos number (3,8,11) as a source of the training data for the RNN is that it is quite hard to extract complete and consistent eye images from these videos without annotation. The face and the eye detectors didn't work properly with these particular videos.

The training of the RNN is performed by dividing the latent sequences to 1200 for training and 200 for validation. After a few trials with different optimizers, the scaled conjugate gradient optimizer (Moller, 1993) showed stable learning after 1000 epochs.

The test phase was performed without any eye image annotation. The software first runs the face and the eye area detectors, and then the obtained eye images are resized to fit the CNN network input. The program passes the left and the right eye images and obtains two readings that make a vector of six elements (3 states confidence levels for each eye image). In case that the eye detector cannot detect the eye area, the CNN will not receive the required input, and therefore the frame will be ignored. The input sequence is buffered and shifted each time the data for the new frame arrives. The new input sequence contains data from the last six frames plus the new frame.

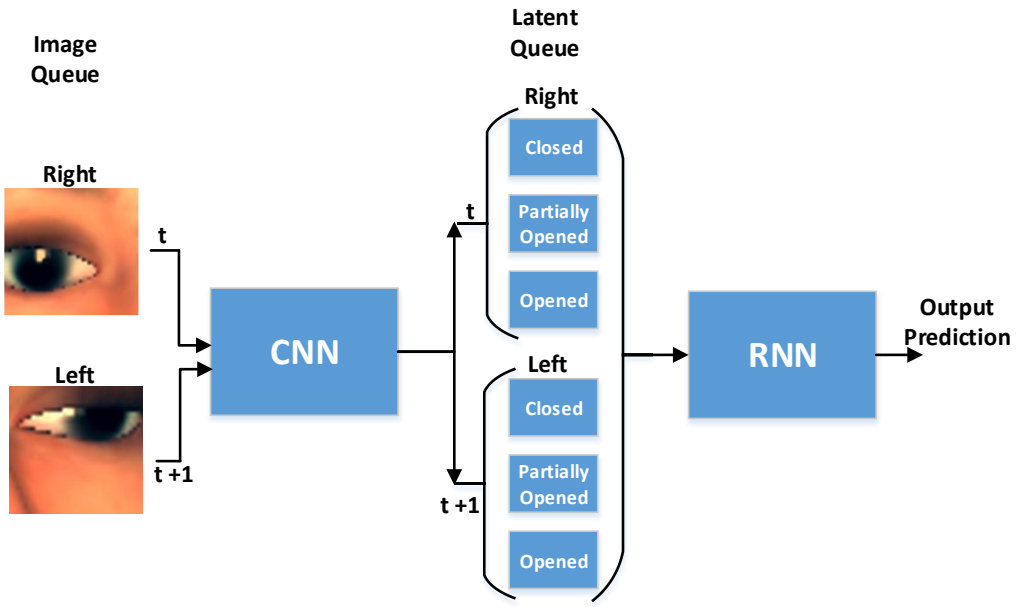



Figure 4.5: The CNN-RNN processing pipeline diagram.

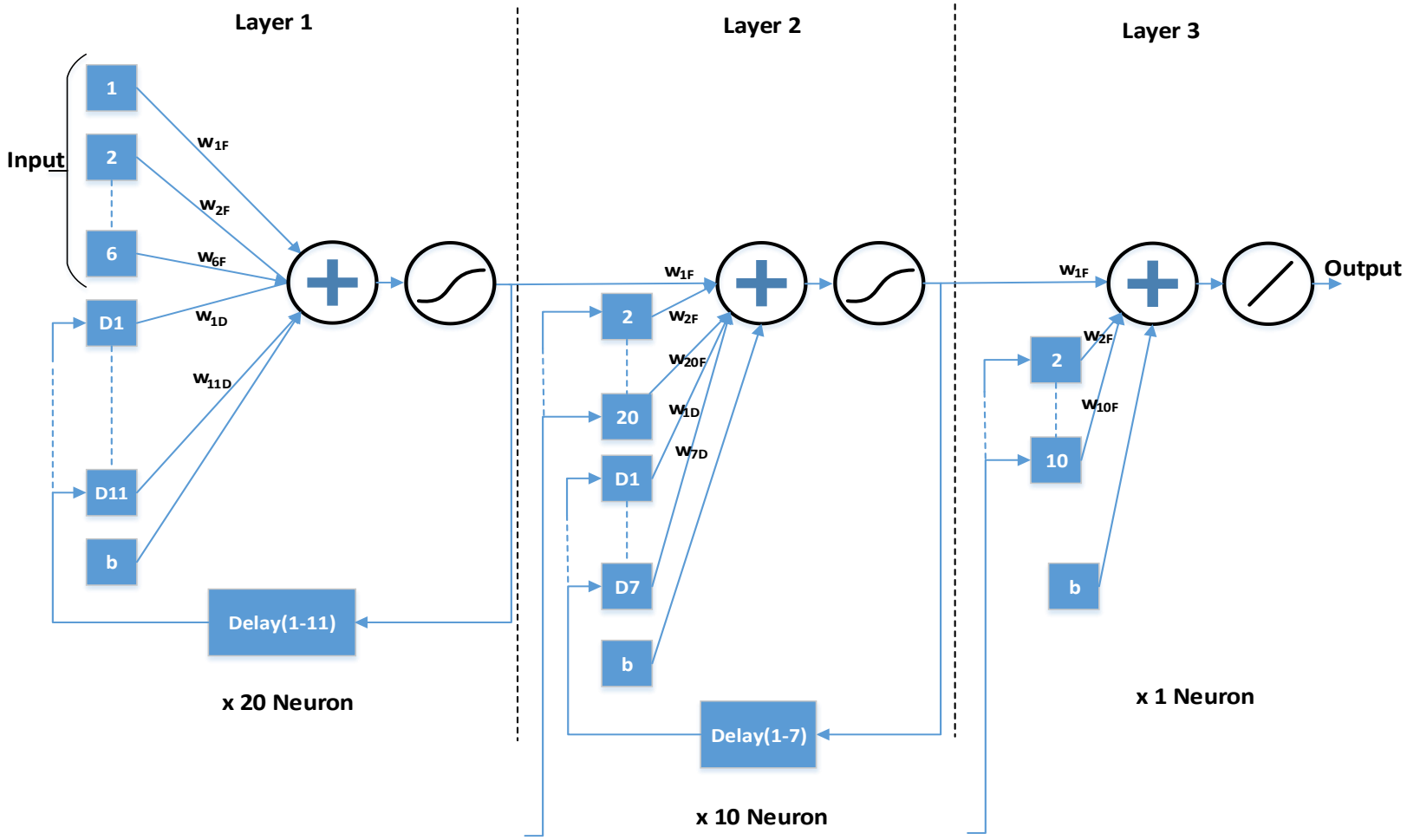



Figure 4.6: Details of the RNN network adopted for the CNN-RNN model used eyeblink detection. The network consists of three layers. The first layer consists of 20 neurons, the second consists of 10 neurons, and the third layer consists of one neuron.

Figure 4.7 shows the ConscisedNet details.

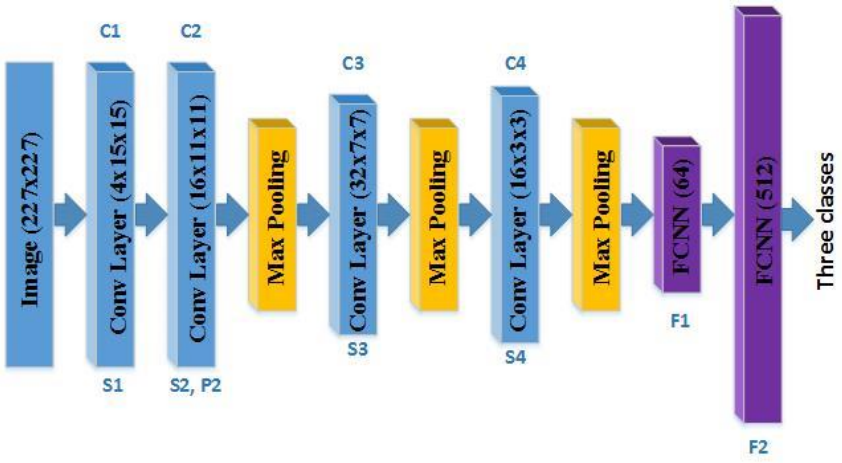



Figure 4.7: The proposed ConcisedNet architecture (Anas & Matuszewski, 2018).

### 4.4.4 CNN-RNN Model Results

Since the RNN model was trained with the ground truth represented by {-1,1} for no-blink and blink, respectively, the model's output is bounded by these values limits. However, the results show that proper thresholding is needed to be selected within the range of {-1,1} that achieved the highest score. To seek the optimum threshold, the F1 score has been employed (Appendix A). The performance of the CNN-RNN method is detailed in Table *4.4*-Table *4.5*. Here, PR: Precision, RE: Recall, GT: Ground Truth, TP: True Positive, FP: False Positive. FN: False Negative (Appendix A). The obtained results show high detection accuracy of the two tested CNN architectures' eyeblink action, i.e., the ConcisedNet-RNN and the AlexNet-RNN. The two networks provided very similar results on the Talking Face dataset of about 97-98% for both precision and recall. However, AlexNet-RNN pipeline offers a better result on the Eyeblink8 when compared to ConcisedNet-RNN. As it has been already mentioned, the implementation of ConcisedNet was tested mainly to address the real-time processing requirements with the software running on a CPU, while AlexNet needs a GPU to work in the real-time regime.

| Dataset | PR % | RE % | GT | TP | FP | FN |
|---|---|---|---|---|---|---|
| **TF** | **97** | **97** | 59 | 57 | 2 | 2 |
| **Eyeblink8** | **75.35** | **84** | 255 | 214 | 70 | 41 |

Table 4.4: Results obtained for ConcisedNet used in the CNN-RNN model for the two datasets, EyeBlink8 and Talking Face.

| Dataset | PR % | RE % | GT | TP | FP | FN |
|---|---|---|---|---|---|---|
| **TF** | **98.3** | **98.3** | 59 | 58 | 1 | 1 |
| **Eyeblink8** | **93.6** | **93.9** | 255 | 241 | 17 | 14 |

Table 4.5: Results obtained for AlexNET used in the CNN-RNN model. It provides better results when compared with the results obtained with the ConcisedNet.

It can be seen that the ConcisedNet performs worse in the prediction mainly due to the size of the network, which is about 10% of the AlexNet size. Here, videos number 3,8, and 11 were used for training.

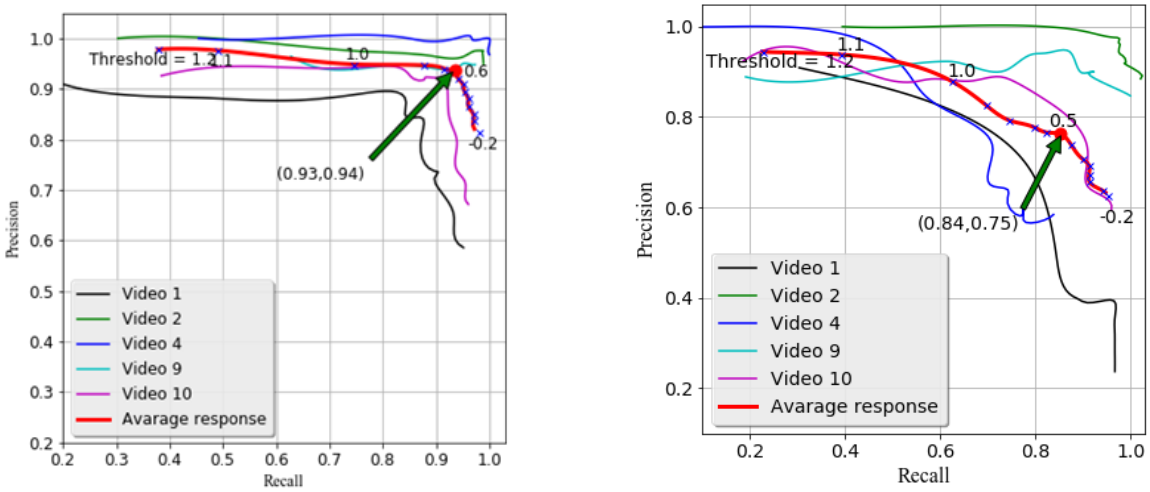



Table 4.6: AlexNet (left) and ConcisedNet (right) for EyeBlink8 with highest F1 score at Threshold = 0.6.

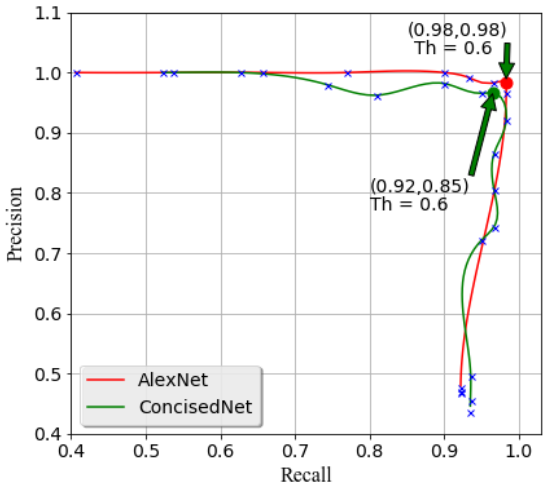



Table 4.7: AlexNet and ConcisedNet, Precision vs Recall for Talking Face dataset. Thresholds = 0.5 and 0.6 provide the best results for the AlexNET and ConcisedNet, respectively.

It can be seen from the comparative results between the two networks that although the RNN trained with sequences generated by the ConcisedNet, the results show that a pipeline with AlexNet-RNN performs better, which suggests that the model is not biased to the ConcisedNet.

In Figure *4.6*, AlexNet-RNN shows better performance of an average of (0.94,0.93) precision and recall at the threshold of 0.6 on EyeBlink8 dataset compared to (0.75, 0.84) precision and recall at a threshold of 0.5. The reason for these low results of the ConcisedNet-RNN is mainly related to the prediction of this network's eye state compared to the AlexNet. The subjects in EyeBlink8 pose with different angles that make the status's prediction of the eye image harder for the ConcisedNet. On the other hand, AlexNet has more capacity to predict the right status, and the model pipeline generally performs better with AlexNet. Meanwhile, the results of the performance are close in the case of the TalkingFace dataset. This performance is mainly related to the high quality of the eye image extract, as well as the subject is not making sudden moves that affect the prediction stage of the eye state.

It is quite important to mention here that some applications could benefit from this type of solution because a spatial inferencing model can provide spatial features. Following the same procedure, the task is to append a time series model to equip the model with temporal inferencing capability with its current capability of spatial prediction (Kornysheva & Diedrichsen, 2014). An example of this case is a trained model to predict the caption from one image to be more expressive by allowing temporal prediction for more details within the caption.

## 4.5 CNN-LSTM Eyeblink Detection Model with End-To-End Training

Training for spatial and temporal features separately is one of the possible options. Also available end-to-end training solutions and achieve the same functionality. Network architectures like early or late fusions (Karpathy, et al., 2014) are specifically designed for Video classification purposes that could be used for eyeblink detection.

In the second proposed eyeblink detection solution, instead of training the CNN and the RNN separately, four different network structures are trained end-to-end using. In addition to the early and late fusions mentioned above, another two networks based on timely distributed CNN are employed. Essentially, time-distributed CNN consists of parallel convolutional channels that sharing weights between them. Across one layer of the CNN, the same weight value appears on the next parallel channel. The weight update is also performed one time per iteration per share weight and propagate across the channels. The output of the Time Distributed CNN (TDCNN) is fed to the LSTM. The input to

these networks is chosen to be five frames at max as it represents half of the average eyeblink duration, which is the eyelid movement from down to up.

### 4.5.1 CNN-LSTM Training

Compared to the previous solution (CNN-RNN method), this solution is trained end-to-end. Furthermore, the dataset that has been utilized for the training depends only on the eyelid motion from down to up, considering the ground truth to be only “No blink” and “Blink” classes of actions. In this case, the network will not depend on the eye states for the training, but it depends on a sequence of gradual transition of the eyelid from the downstate to the upstate. Only for eyelid transition from the downstate to the upstate will the model provides a true blink state at its output. Any other sequences are considered as no blink. Therefore, the network should be able to capture that transition, whether the blink action is complete or not complete, short, or extended as long as the proper transition in the eyelid is captured.

To obtain a successful action completion solution, the training dataset needs to reflect the action in complete form without losing the generalization of the problem. In this case, the video for training should contain all or most of the action snaps. The length of the action and its complexity are the main factors affecting the length of the training video.

Previously, it has been mentioned, that a blink action could takes about 12 frames images in a 30f/s video. The assumption that could proposed here is that instead of training a model for a complete cycle of the eyeblink, it could be easier to train or part of the action, which is the part that shows the completion of the action. Thus, instead of 12 frames, five frames can be used only to show an eyeblink by capturing the eyelash state from down (eye closed) to up (eye open). Samples of representative training sequences are shown in Figure 4.8 that reflect sets of action completion for the eyeblink action. As it shows, sequences like all opened eyes, partially opened eyes and closed eyes, or any other mixed combination are all to be considered as “No Blink.” The only sequence that is considered a “Blink” is as shown in the last row in the figure, where the sequence starts with a closed eye and ends with the gradual opening of the eye. The same dataset that are used previously to train AlexNet and ConscisedNet for classification are used for training. Since the dataset is organized as three subsets for ‘Open’, ‘Partially Open’ and ‘Closed’ with 1000 images per set or class, the training program randomly pulls images from these three subsets to form balanced “Blink” and “No blink” training sequences. The first 800 images from each subgroup are assigned for training, while the other 200

are used for validation. The testing is performed on the dataset video (Eyeblink8, TalkingFace, and ZJU).

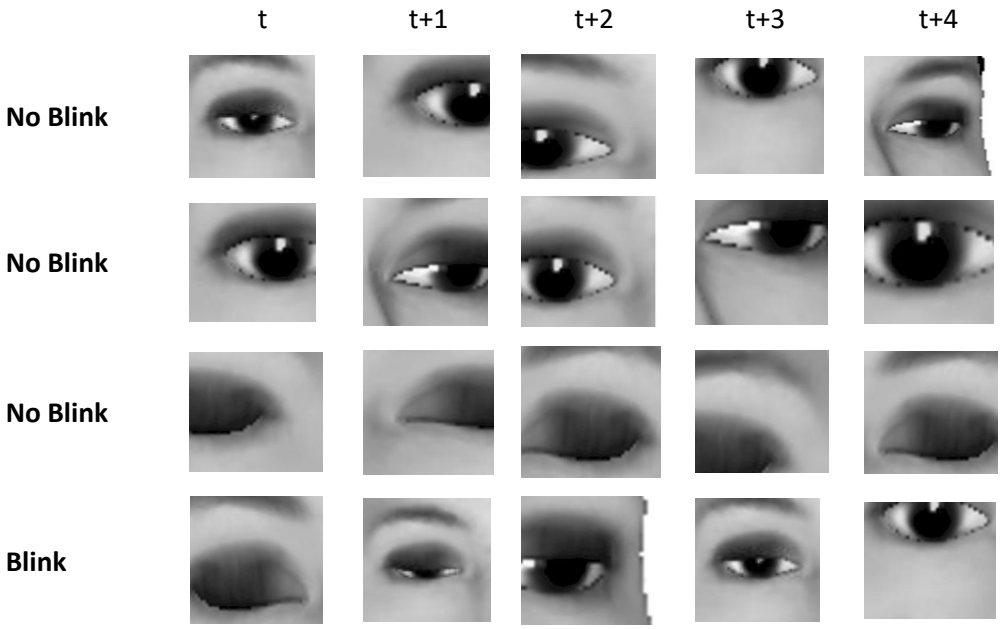



Figure 4.8: Representative sets, showing five frames of training sequences. The first three rows do not represent an eyeblink; The last row depicts an eyeblink that consists of one closed eye followed by partially open and opened eye images.

In addition, Figure 4.8 shows the eyes in the images are not centered in the middle of the image to which demonstrates the real performance of an eye detector operation. Such a situation does not comply with the requirement for the motion detection problem firmly. For the typical motion detection problem, a model may capture the variation in the image's intensity in the same area that appears in the first frame and continues through the frame sequence. The third row represents such inconstancy to the motion detection requirement or constraint. However, the requirement of this case is to address this extremely weak correspondence issue using the DL model. Also, the generated sequences contain a mix of left and right images to simulate the case when the eye detection model detects only one eye within the face area.

### 4.5.2 Networks Implementation and Comparisons.

Eyeblink is a temporal action in video that represents a sequence of events occurring during the motion of the eyelids downward then upward, having the opened eye as a default state. In this part of the implementation, different scenarios will be tested to address the problem and provide the related results. Figure 4.9 shows four models trained for different input video sizes using in the first two experiments, two scaled-down models from (Karpathy, et al., 2014), the early and late fusions models.

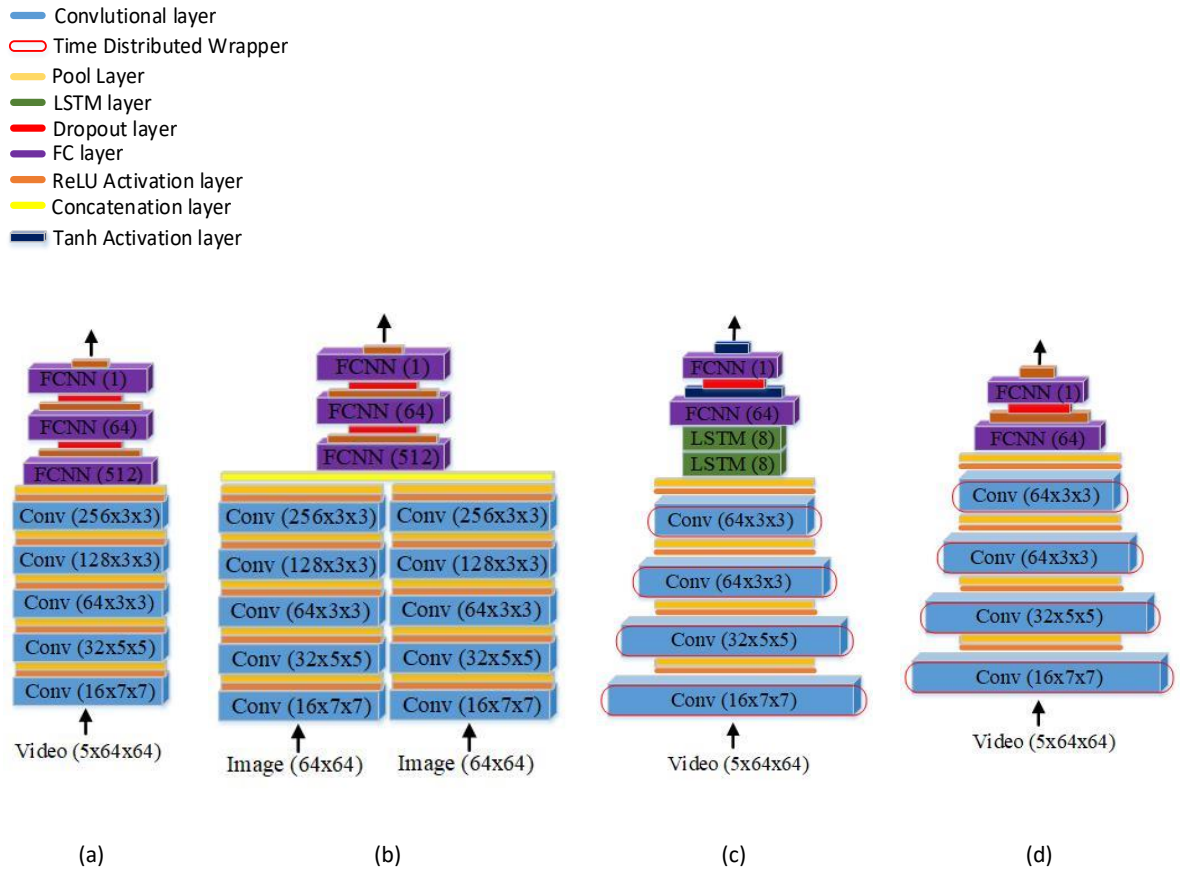



Figure 4.9: Suggested models for Eyeblink Detections. a) Early fusion with five frames instead of 2. b) Late fusion (Karpathy, et al., 2014). (c) Proposed network time-distributed convolutional network with LSTM, (d) The same network as in (c) but without LSTM.

For all the networks, the same training setting is applied and trained end-to-end. The input to the network is different only for late fusion, consisting of two images, while the other models are trained with the same input size of 5x64x64. The models themselves are not the same, but the focus here is whether TDCNN and LSTM can perform better for this type of problem. To train these networks, Mean Square Error (MSE) is used to measure the loss, and Adadelta is used as an optimization algorithm for all the models (Wang, et al., 2019).

The time distributed layer is essentially a wrapper layer that replicates the weight without explicit configuration required for weight sharing (Appendix C). Thus, the five input frames are passing through the same weights. The convolutional layers within the model are wrapped with a time-distributed process to allow sequential input from each convolutional channel. The five channels represent one-time step to the LSTM. The model's convolutional layers start with a kernel size of 7x7 pixels working on a 64x64 image. The kernel sizes are reduced to 5x5, 3x3, and 3x3 to fit the feature space dimension, while the maxpool layer reduces the data spatial representation dimensionality to half after each convolutional layer until it reaches the LSTM layers. The time-distributed convolutional

layers accommodate variations in the eye image like scaling, translation, orientation, and pose while keeping the temporal representation within the feature space to the next layer. This wrapper helps implement a sequential form of the convolutional layer. Each frame of the input video produces its feature vectors, and the new features are passed sequentially in the same way to the later stages. Then, the layer's output is passed to the next layer to start the same process until it reaches the LSTM layer. At the LSTM layer, the input is transformed into a 1D vector to train the layers accordingly. The 1D vector represents the feature vector generated from the video of the five frames. At this stage, the vector size is 5x4x4x64, where each frame is represented with a tensor of 4x4x64 feature size. Then, the features flatten to construct a suitable input to the LSTM layer of eight cells per layer.

### 4.5.3 CNN-LSTM Results

The results obtained by training the four models of Figure *4.9* are shown in Figure *4.10*, which shows a better performance achieved by the time distributed network tested on the EyeBlink8 dataset.

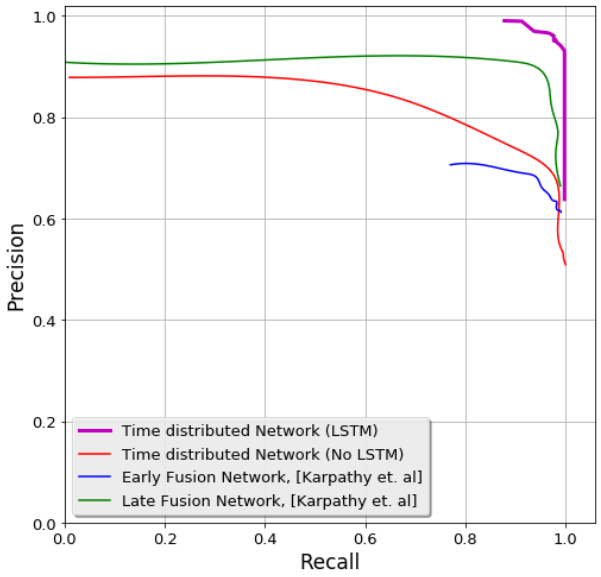



Figure 4.10: The network performance of Figure (8) tested on the Eyeblink8 dataset. The high performance obtained with time distributed CNN with LSTM layers.

Figure 4.10 is shows that late fusion performed better than the early fusion and the time-distributed CNN (TDCCN) without LSTM, although the TD and early fusion input use five frames compared to only two frames of the late fusion network. However, when the TDCCN is equipped with two layers of LSTM, the performance of the network increases. This experiment shows that for each case that needs to be optimized, a different number of hyperparameters need to be investigated to reach the highest performance.

Additionally, it had been noticed that adding the Tanh activation layer to the end of the network provided better results than using Relu activation. This could be related to the fact that the previous layer is an LSTM with output activated by a sigmoid function that introduces a nonlinearity, especially near the decision boundaries. Having a Tanh at the FCNN and train the network with a {1} and {-1} ground truth values increased the dynamic range and exploited these nonlinearities and provided better inferencing. The same case didn't help with other networks that they did not implement an LSTM.

To optimize the TDCNN with respect to the number of the LSTM layers, three options were tested. Figure (4.11) shows an experiment performed after training the best-performed network from the previous step with different dimensions and the LSTM layer incorporated. The figure shows the results when five frames are used with two LSTM layers. Clearly, the effect of adding a second LSTM provides marginal improvement compare to adding more frames. Of course, different scenarios can be tested here, but the focus of is experiment is to show the effect of the number of LSTM and the input size on the network performance.

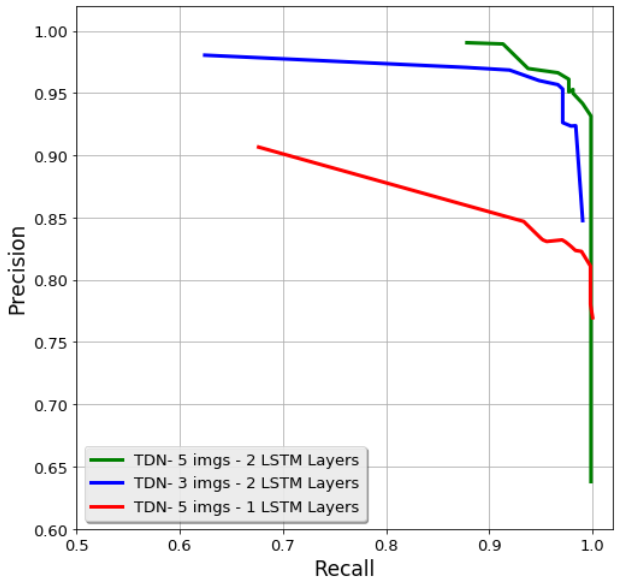



Figure 4.11: The networks performance of Figure 4.9(c) tested on the Eyeblink8 dataset. The highest performance is obtained with Time Distributed Network with two LSTM layers when the input has five frames. The network performed less when the input is three frames or when one LSTM layer is used.

The experiment of Figure 4.11 shows that having two LSTM layers increases the network's performance while the number of images contributed less to the performance. Thus, the two layers LSTM network after the TDCCN shows better inferencing compared to the one layer. This result could mainly relate to the level of the non-linearity that the network layer introduced to the model. It is

worth mentioning that adding more LSTM layers does not improved the performance but, on the contrary, reduced it to lower than the two LSTM layers network. Also, storing action information concerning the problem at hand could be better presented with the two layers LSTM when the input size was five frames. In the meantime, having a model with the same setting and adding another LSTM layer to have three LSTM layers wasn't perform well, which could be mainly related to the increase in the complexity that generates more error to the network and becomes hard to train. Therefore, the experiment stopped investigating more network configuration.

The capability of the Figure 4.9(c) model to process spatiotemporal information starts early in the first convolutional layers until the layers before the LSTM. Figure *4.12* shows one feature map in different layers of the model before the LSTM layers. In the static case, when an image is applied to the classifier network, the resulting feature map usually has crisp edges representing borders and the object's orientation, as shown in Figure 4.3. In this case of Figure 4.12, the feature maps are blurred, reflecting motion properties in the case of shooting a moving object with a camera.

It can be seen that the representation between the open eye and closed eye images appears more clearly at the end of the convolutional layers and before entering the LSTM layer; see Figure *4.12*(e). Before the LSTM layer, the motion sequence's representation is compressed by the feature maps and presented to the LSTM layer. This stage's feature set should produce sequences representing a different variation of the eye-opening within the video sequence. It can be seen that that for the opened and closed sequence, the feature is quite different.

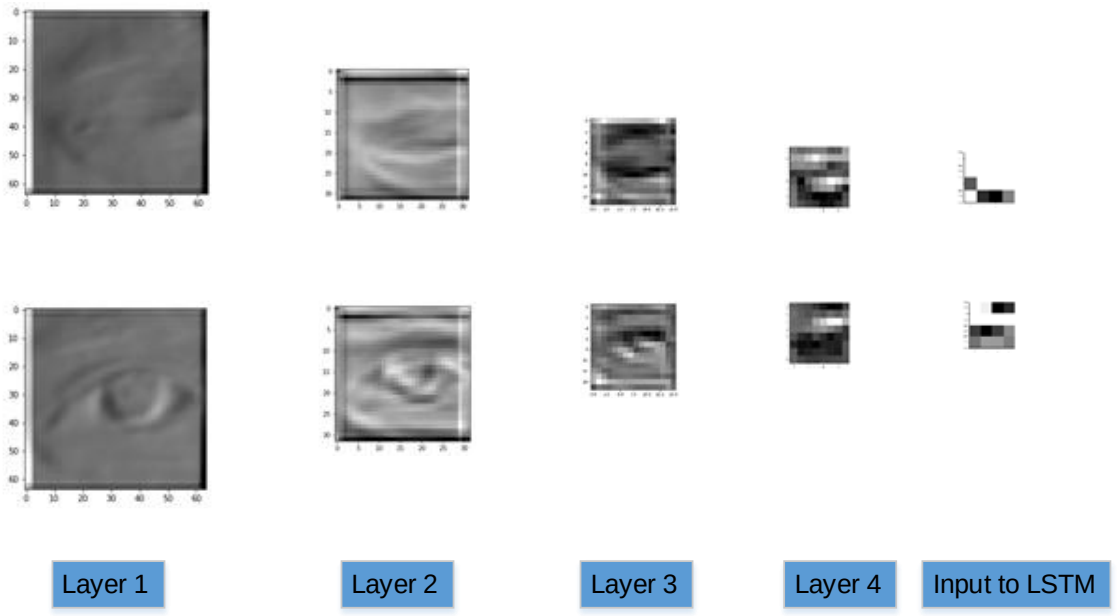



Figure 4.12: Feature maps for Eyeblink Detection. Two images, one for a closed eye (Top) and another for an open eye (down), are used to visualize eyeblink detection. The developed kernels blur the images, which shows a similar effect when the camera captures a moving object. The final layer, which represents the input to the LSTM, shows a clear distinction between the two patterns. Note that the images are magnified for clarity.

Figure 4.13 shows five timesteps of the Blink sequence and another timestep of the No-Blink sequence. Each row of this timestamp passes to one LSTM cell, and the data shifts eight cells. The output of the LSTM passes through the time distributed drop out layer to reduce the overfitting and improve the training process, and then the data flatten to inter to the FCNN.

The two sequences of the figure show a significant difference during the feature's development through distributed layers. The distinction appears through the activation pattern in the blink action, which is the eyelid transition from downward to upward compared to the case when the eye is open.

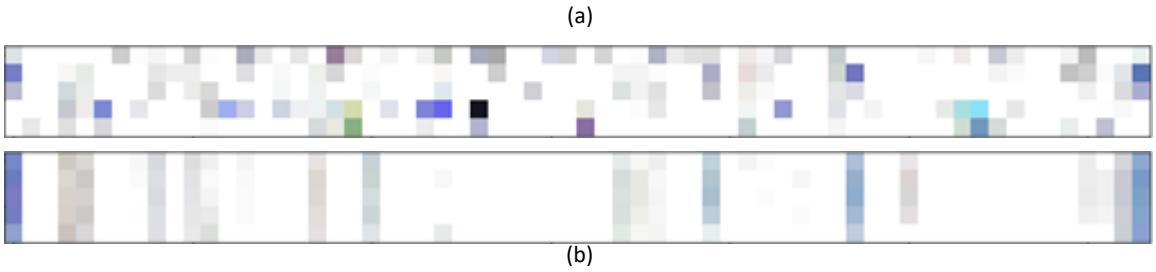



Figure 4.13:The feature vector generated at the last convolutional layers represents the input to the LSTM layer. (a) Represents the activation when a video containing the eyelid transition is applied (Ture Blink). (b) Represents the activation for a video containing open eye frames.

### 4.5.4 TDCNN-LSTM Results

In this work, the model outputs value between [-1,1] based on the range from the model's final Tanh activation layer. Therefore, the requirement is to provide a threshold that can be used to decide whether the model's output is an eyeblink or not. Figure 4.14 shows the network's instantaneous response to a video applied from the Talking Face benchmark. Both, the ground truth and the model output value have been normalized to provide better visual comparison between then. It can be seen that the ground truth areas when the eye is close. The model's response is delayed for some frames. The model needs to receive the complete sequence that represents the transition, which must include one downward eyelid (closed eye image) followed by the rest transitional eyelid images states toward the opening.

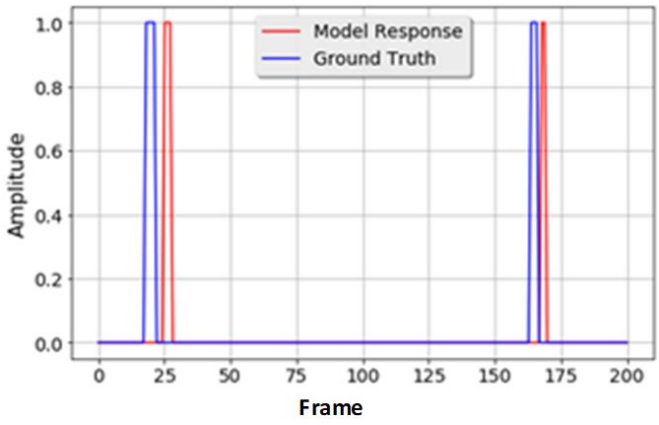



Figure 4.14: Model response to the input from the Talking Face dataset. The response of the model lags by a few frames. As the video buffer receives the targeted blink sequence, including the eyelid transition, it responds within a range of 2 to 5-frame delays. During the evaluation, this delay is considered a correct response and compensated for during the performance estimation.

To generalize the model, different eye shapes with varying tones of colour and illumination are incorporated to augmenting the sample acquisition data. Also, an image sequence may contain eye shapes for different subjects. Images of different eye shapes were organized into distinct sets of videos

of five frames each. The non-blink videos included frames of all open, partially open, and all closed sequences of image. In this case, the focus is on the true blink pattern, which consisted of videos starting with one frame when the eyelid is downwards (eye closed) and followed by frames of the eyelid upwards eye open (or partially open). The choice of the five frames per video was generally made after visual evaluation of eye shape during the transition period. The choice of five frames' length ensured the transition that had been occurred was sufficient.

**Error! Reference source not found.**,16,17 show precision vs. recall for the three datasets reviewed in S ection (4.5.2) using the proposed TDCNN-LSTM model. Precision and recall are estimated for different threshold values in steps as shown, and then the F1 score is used to estimate the best precision and recall and their corresponding threshold values.

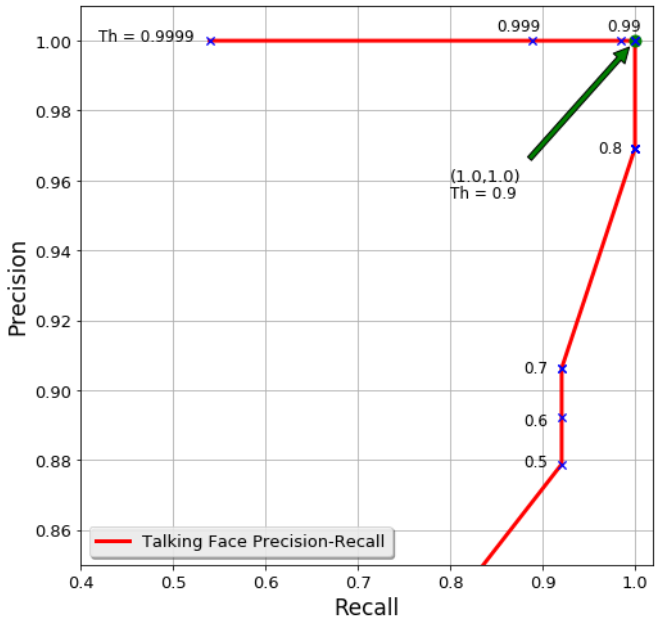



Figure 4.15: Precision-Recall curve for the TDCNN-LSTM model on the Talking Face dataset. The model performs best at a threshold of 0.99.

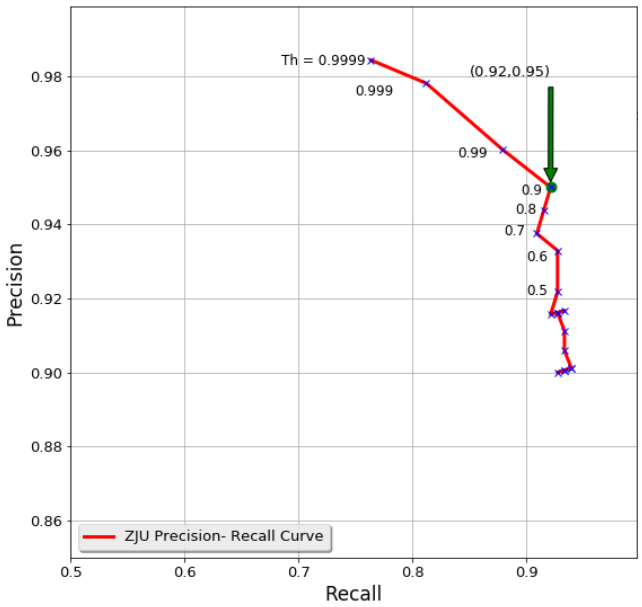



Figure 4.16: Precision-Recall curve for the TDCNN-LSTM model on the ZJU. The model's best performance is obtained at a threshold of 0.95.

The low performance in the case of ZJU is mainly related to two issues. First, the eye detector did not perform appropriately on these low-resolution videos. In many cases, the eye images cropped after the detection operation was half-eye images or images of another part of the face, which affected the prediction stage and led to many false positives or false negatives. Second, in other cases, the eye detection stage could not detect an eye area, explicitly losing the detection capability when the eye was closing, which led to false negatives. In the latter case, when the eye detection stage was unable to detect the eye area, the ground truth and the prediction stage were bypassed, which led to a lower ground truth of 165 blinks for this dataset.

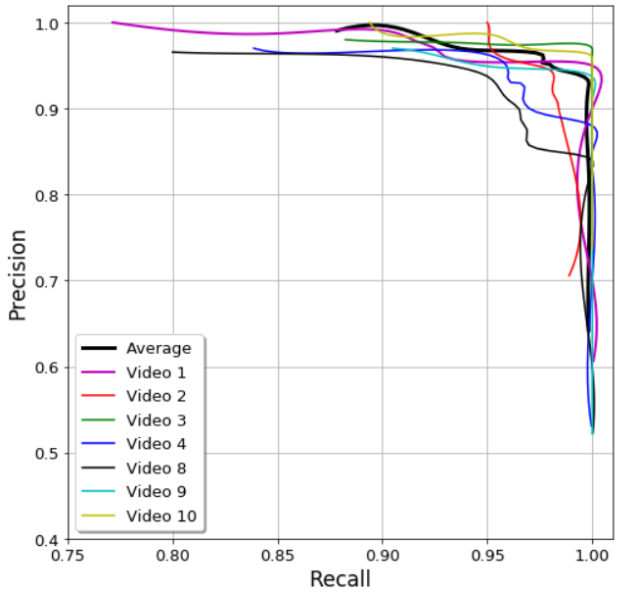



Figure 4.15: Precision-Recall curves for the TDCNN-LSTM model on the Eyeblink8 dataset. The highest values of precision and recall are obtained after estimating the F1 and the corresponding value of the threshold is applied. In this case the threshold value is 0.95.

Figure 4.17 shows the performance of the TDCNN-LSTM network to the Eyeblink8 dataset. Each curve represents the Precision/Recall for that video.

Table 4.8 lists all the three datasets' results, where the F1 (Appendix A) score was the highest.

| Dataset | PR % | RE % | GT | TP | FP | FN |
|---|---|---|---|---|---|---|
| **ZJU** | 95.0 | 92.1 | 165 | 152 | 8 | 13 |
| **TF** | **100** | **100** | 63 | 63 | 0 | 0 |
| **EyeBlink8** | | | | | | |
| **Video 1** | | | 35 | 32 | 0 | 0 |
| **Video 2** | | | 86 | 83 | 4 | 3 |
| **Video 3** | | | 51 | 50 | 2 | 1 |
| **Video 4** | | | 31 | 30 | 3 | 1 |
| **Video 8** | | | 31 | 30 | 2 | 1 |
| **Video 9** | | | 42 | 42 | 0 | 0 |
| **Video 10** | | | 66 | 64 | 2 | 2 |
| **Eyeblink8** | **96.2** | **96.6** | 342 | 214 | 11 | 8 |

Table 4.8: Tabulated results obtained for the TDCNN-LSTM model for all benchmark datasets. In the case of Talking Face and Eyeblink8, the model provides high detection capability. (PR: Precision, RE: Recall, GT: Ground Truth, TP: True Positive, FP: False Positive, FN: False Negative)

The model's average response to this dataset is shown in Figure 4.18, and the result of the model is compared to the previously reported work.

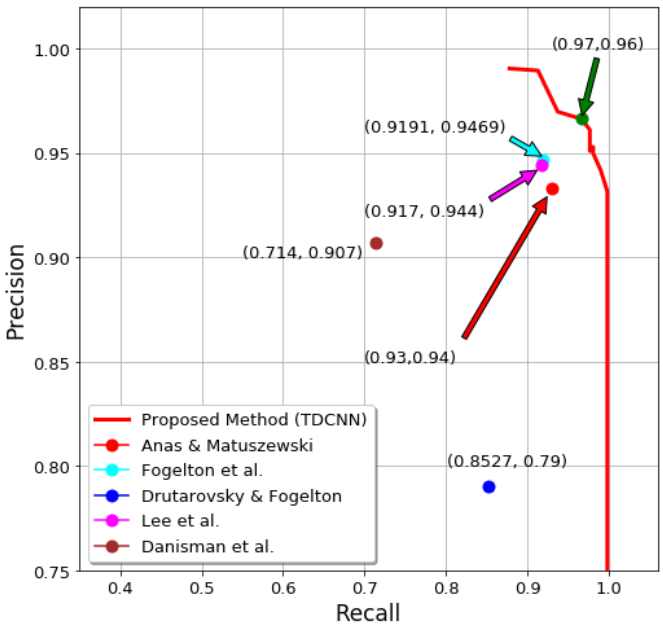



Figure 4.16: TDCNN-LSTM model average performance on Eyeblink8 dataset compared to other methods reported in the literature. The proposed method outperforms existing, state-of-the-art methods.

Figure 4.19 shows the model response to the TalkingFace dataset. The model showed full precision and recall responses for the dataset, mainly due to the excellent quality of the image provided within the dataset, allowing good detection from the eye detection step. On the contrary, the results for ZJU in Figure 1.20 were low compared to previously reported work, and the main degrading reason was the detector performance on these low-quality images.

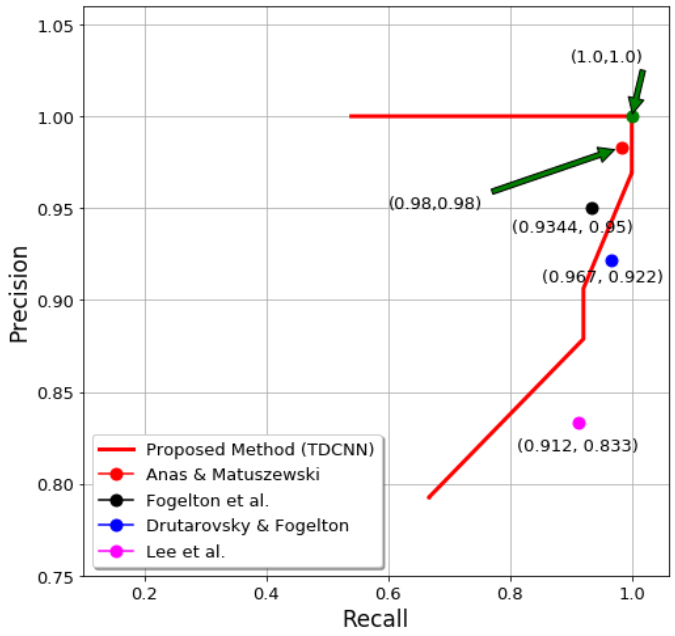



Figure 4.17: TDCNN-LSTM model performance on TalkingFace dataset compared to other methods reported in the literature.

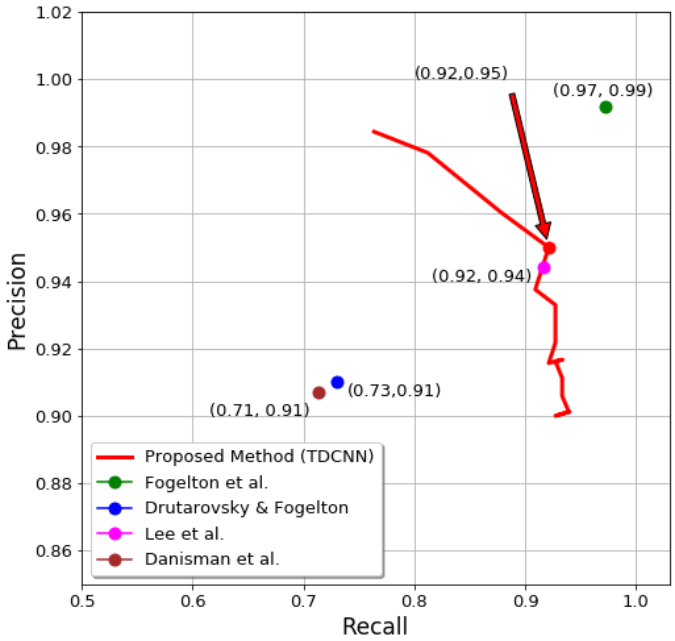



Figure 4.18: TDCNN-LSTM model performance on ZJU dataset compared to other methods reported in the literature.

The two main features of Eyeblink8 and Talking Face are that the subject's face occasionally moves. While this could be a problem in other methods that depend on the motion vector as a tool to detect the blink action, it did not affect our method even when the CNN received a blurred eye image resulting from movement.

In the field of Human-Computer Interaction, eyeblink can represent an easy way to communicate with computers and command them to execute specific tasks (Królak & Strumiłło, 2012). It has been suggested that a double blink could be used to represent a mouse click. In the next experiment, the granularity of the model with double blinks is discussed. Figure *4.19* shows an experiment to test the network's response to a double blink appearing in a sequence.

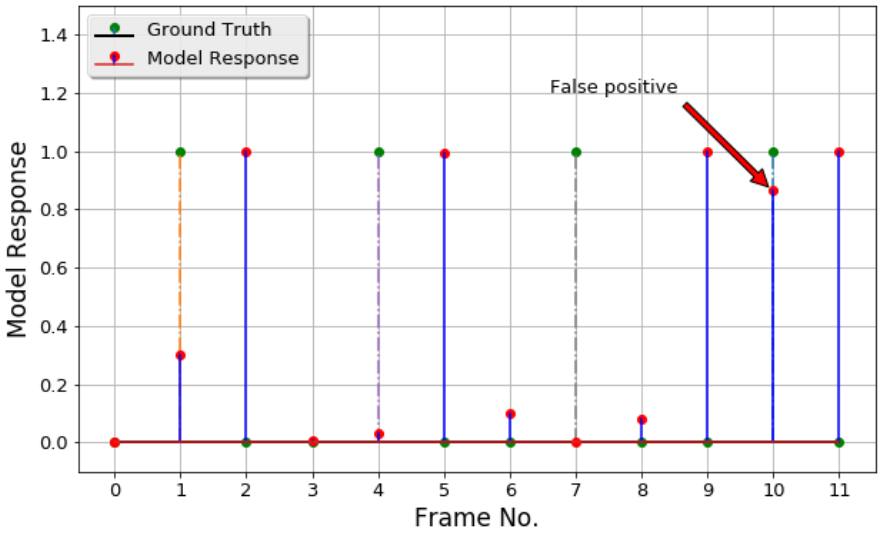



Figure 4.19: Evaluating a double blink in the model's response. On consecutive blinks, the model produces false positive.

Here, independent video sequences are shown with ground truth in green. The model responds almost immediately to the very next frame for ground truth 1 and 4. In the case of ground truth on index 7, the model delays two frames to respond on index 9. It takes a very short time for the model to reach its minimum value when the next ground truth appears in index 10. If we consider the threshold to be 0.6, then five blinks will appear instead of four. Based on this observation, the model could respond to double blinking after three to four frames with the eyelid open following the last blink.

## 4.6 CNN-RNN vs TDCNN-LSTM

Now comparing the two methods of training both models, the performance of the TDCNN-LSTM is higher for the three benchmark datasets. The way how the two networks process the images are different. While the CNN-RNN depends mainly on the spatial information to estimate the eye state, the TDCNN-LSTM process videos consist of 5 frames and develop spatiotemporal features early in the

network. Additionally, the training of the CNN-RNN is performed in two steps and on 1200 sequence for the RNN, while the training of the TDCNN is performed end-to-end and on a much higher dataset.

Another important finding of this experiment can be deduced from the data type used in this motion detection problem. As shown, the dataset used for both methods contain image sequences belonging to different people and without clear correspondence between the image from one frame with respect to the next frame's image. Figure 4.8 shows that although the eye area is not centred in the image, in different scales, and even cut within the sequence, the models perform with high performance, especially when trained end-to-end. The dataset's specification implicating that the model can develop proper correspondence in the sequence as the data flows into the upper feature sets and when they reach their compressed representation in the feature set.

## 4.7 Summary

In this chapter, real-life motion detection of the eyeblink problem has been approached with two different formulations. The problem was formulated first with a complete sequence of a blink to represent a complete short blink cycle of seven frames. In the second formulation, an action of the eyelid moving from down to up is considered as an indication of a blink within five video frames. Two deep learning models developed for eyeblink detection are trained with varying training schemes and tested on three popular publicly available datasets. The first model consists of CNN followed by RNN, each trained separately such that the RNN network is not biased to a specific CNN model. For this case, the RNN model's trained data obtained by a CNN model coined ConcisedNet and the test performed for both AlexNet and ConcisedNet. The trained RNN model shows unbiased property and outperforms the TalkingFace dataset. The second model is the TDCNN-LSTM trained end-to-end and outperforms on TalkingFace and Eyeblink8 Dataset. Results show the TDCNN-LSTM network achieved ZJU: 95%,92.1%, Talking Face: 100%,100%, and EyeBlink8: 96.2%, 96.6% precision, and recall.

The TDCNN-LSTM model showed better performance of any type of network and solution previously reported. The network shows an early response to the motion in the most initial layers of the network, and the representations at the LSTM layer show a clear distinction between the “Blink” and the “No blink” classes. Neither using one LSTM layer nor three LSTM layers improve the performance of the TDCNN-LSTM based on two LSTM layers. For the prior case, one LSTM apparently did not provide the necessary complexity of the network boundary to detect the right sequence. At the same time, the latter increases its complexity to a level that the training became too complicated and the performance degraded. Using two layers of LSTM for this motion detection problem showed an

optimum complexity to provide the network's best performance. Many researchers used blink detection in many applications has been reported in the literature mentioned previously. Having a the proposed robust and practical approach in this work will enhance these studies. Many of the natural activity by a human or any other phenomena could be formulated within a time sequence and approached in the same concept discussed in this chapter.

## 4.8 Contribution

In this chapter, two formulations of the eyeblink detection problem have been presented. Both problem formulations and the proposed corresponding deep models are novel. In the first model, a ConscisedNet and AlexNet have been used as spatial to latent features extractor, and an RNN network was implemented for temporal features extraction. Each of the ConscisedNet-RNN and AlexNet-RNN performance have been tested for different eyeblink detection datasets. The experiments showed that the RNN is not overfitting to a specific spatial to latent information present in the dataset used for training but is capable of responding to the time domain's data profile. The second developed model suggests that the action's final part could only be utlised for an action to be considered complete. A novel TDCNN-LSTM model has been developed for this task. The model shows early motion related features developed in the model, reflecting the model's capability to address more complicated motion detection tasks. The model shows the outstanding performance on the giving task.

# Chapter 5 - DISPARITY WITH CNN

## 5.1 Introduction

In Chapter 2, the 2D motion concept has been introduced. It has been presented that the optical flow is a 2D motion estimation with unrectified stereo image pair constrained. A special case of the 2D estimation appears when the stereo image pair rectified. This particular case is referred to as a disparity estimation. All the equations applied for the 2D motion estimation are applied on disparity estimation, and the same DL model could be used for optical flow or disparity or both. In this chapter, the incorporation of the occlusion in enhancing the disparity estimation (Hirschmuller, 2006) during the model training is presented. The chapter includes the analysis of the occlusion effect on assessing the disparity map self-supervised learning process. The method of estimating the occlusion will be followed, including examples that shows how to utilize occlusion for better representation of the loss function using occlusion masks. Then, a mathematical calculation will be explained in companion with the implementation details of the occlusion mask. The chapter also presents the suggested network implementation detail with the mask estimation module attached to the network. The loss functions enhanced by the occlusion mask capability implemented are also explained.

Furthermore, the datasets that are used in the evaluation phase with their related specification and the augmentation techniques employed are described briefly. The qualification and the quantitative results are reported at the end of the chapter using the datasets' test set for this particular problem. Then the related analysis of the results will be present.

## 5.2 Related Work

Many algorithms have been proposed to address the issue of disparity estimation. In the listed methods below, the consideration is that the two images are rectified, and the epipolar constrain is satisfied. The main techniques used nowadays include the following:

1. Feature-based methods: As the name suggests, feature-based disparity estimation depends on finding the features between two images that represent sparse disparities and then involve other algorithms to produce dense representations (San & War, 2017).
2. Block-based methods: Here, the correspondence for a block area in one image is compared to some blocks in the second image's shifted regions.

3. Phase-based methods: Based on the Fourier shift theorem, the phase formation between images can be translated to correspondence, and the disparity can be estimated from the representation of the Fourier-phase image between the left and right images (Li, et al., 2016).
4. Deep Learning-based method. In this method, a combination of many techniques, including the above-mentioned techniques had been utilized (Mayer, et al., 2016).

### 5.2.1 Feature-Based Method

There is a great deal of literature dealing with matching feature points and edges under various conditions covered in this method to estimate disparity. This method mainly provides a sparse representation of the disparity. Scale Invariant Feature Transform (SIFT) feature or similar feature-based matching algorithm can be utilised to estimate this motion property as a sparse representation. Other algorithms, like Random Sample Consensus (RANSAC), can also provide disparity reading in a sparse form (San & War, 2017).

### 5.2.2 Block-Based Methods

In the block-based method, a point $p(x, y)$ with its surrounding pixels forming a set called reference block in one image. Assuming that the binocular stereo pairs are rectified and comply with the epipolar constrain, then the candidate block can be represented as $CB(x + d(x, y), y)$ where $d(x, y)$ is the possible disparity. The block size is $(n + 1) \times (n + 1)$, and the searching range of $-d_{max}$ to $d_{max}$. Thus, the block matching can be performed by estimating the maximum value of correlation along the horizontal line where the point $p$ is located. Or the minimum amount of the Sum of Squared Difference (SSD) (Tao, et al., 2008) as in Equation (5.1).

$$E_{SSD}(d) = \sum_{x,y \in \mathcal{B}} [I_l(x, y) - I_r(x - d, y)]^2 \tag{5.2}$$

### 5.2.3 Phase-Based Methods

Phased-based method of disparity estimation depends on the Fourier shift theorem, which states that the delay between the same signal in the time domain is equivalent to a linear phase difference between the Fourier transform of these two signals in the frequency domain. Thus, in a stereo pair of images, the right image translation with respect to the left image can be considered as a linear phase

difference in the frequency domain. This phase difference is a dense disparity map of the image pair (Li, et al., 2016).

The stereo image disparity can be obtained in this method as follows:

$$I_r = I_l(x - d(x, y), y) \tag{5.3}$$

The task here is to find the value of each pixel's disparity with respect to the left image. The Fourier representation of Equation (5.3) can be written as:

$$\mathbb{I}_r(\omega_1, \omega_2) = \mathbb{I}_l(\omega_1, \omega_2).\exp(-i\omega_1 d) \tag{5.4}$$

Multiplying with the complex conjugate of $\mathbb{I}_l(\omega_1, \omega_2)$, the normalized phase correlation can be expressed as in Equation (5.5):

$$\exp(-i\omega_1 d) = \frac{\mathbb{I}_r(\omega_1, \omega_2).\mathbb{I}_l(\omega_1, \omega_2)^*}{|\mathbb{I}_r(\omega_1, \omega_2)|.|\mathbb{I}_l(\omega_1, \omega_2)|} \tag{5.5}$$

Here, taking the Fourier Inverse of the above equation represent the impulse response, which is the disparity value at point $(x, y)$.
Usually, the calculation of the disparity in this method is obtained locally by assigning a window modulated with Gaussian or Gabor kernel for better phase representation in the frequency domain and better quality of the recovered information.

### 5.2.4 Deep Learning Disparity Estimation

Disparity as other motion estimation depends on a pair of images to be estimated. It is still challenging to develop a dataset containing both the image pair and their disparity ground truth. Such a task is expensive and cumbersome to create in practice. This limitation drives the researchers to find unsupervised learning solutions to estimate the disparity and other related motion estimation problems.

Deep learning is a relatively new arrival to disparity estimation using Convolutional Neural Network (CNN). The Network first needs to be trained using a vast number of samples representing almost the entire domain that the model could see during the inferencing operation. The first implementation of the disparity-based CNN appeared in (Mayer, et al., 2016). The implementation of the network encompasses most of the techniques used for disparity and optical flow estimation. For instance, the

network is based on developing features for disparity estimation. It uses the correlation layer (Dosovitskiy, et al., 2015), which performs matching functions that produce features with high responses at the matching locations. In addition to that, the network contains a multi-scale technique to accommodate higher disparity value estimation. Furthermore, the training includes reducing the distance in the measurement between the model's prediction and image disparity ground truth in a supervised learning paradigm.

In general, the model of (Mayer, et al., 2016) consists of a contractive network followed by an expanding network. The contractive part consists of convolutional layers with strides divides the feature spatial dimensions by two at each layer. On the other hand, the expanding part consists of transposed convolutional layers that recover the feature dimension to the original input image size.

Deep learning solution for disparity achieving very successful careers. The solutions tackle highly challenging tasks in disparity estimation. Nowadays, disparity estimation benchmarks, including very challenging characteristics. For instance, outdoor images with extreme variation in intensity and illumination, shadow as in Kitti dataset (Menze & Geiger, 2015) became available.

The implementation of the disparity estimation taking several forms, (Luo, et al., 2016), in a DL model implemented to predict the disparity pixel using ground truth. A small left patch equal to the receptive field of 9x9 pixels of the network from the left image first extracted at random with its companion ground truth. The batch is fed to one branch of a Siamese network while the right image is provided with its original size to the other branch. The outputs of the two branches consist of 64 elements for the left batch and $|Y_i|$x64 for the right branch, where, $|Y_i|$ represents the possible disparity value. The two vectors are multiplied using inner-product to compute the score for each value of $|Y_i|$ disparity and a SoftMax is calculated for all the pixels over the possible disparity. They use cross-entropy loss with respect to the model weights as:

$$Loss_{CE} = \min_{w} \sum_{i,y_i} p_{gt}(y_i) log\ p_i(y_i, \boldsymbol{w}) \tag{5.6}$$

where $p_{gt}(y_i)$ is smooth target distribution centered around the ground-truth $d_i$. After completing the prediction process, post-processing is required to obtain the disparity map. The post-processing, including smoothing using Markov Random Fields (MRF) with other different smoothing strategies.

Convolutional neural network shows a lot of success in many machine learning tasks. In companion with many recognition tasks, CNN became the default solution for per-pixel recognition which gain a lot of attention in the research community. Applications like depth estimation from a single image

(Eigen, et al., 2014), disparity estimation (Zbontar & LeCun, 2016), optical flow estimation (Dosovitskiy, et al., 2015) are a few of many applications where the CNN provide superior solution compared to previous techniques.

Other researchers developed the Deep Match (DM) architecture (Revaud, et al., 2016) utilizing the U-CNN topology. They considered the DM is similar in structure to the CNN network with the benefit of the backpropagation feature that the DM structure is lacking.

It has been mentioned that training DL models requires a big dataset with their ground truth in supervised learning. Authors in (Tonioni, et al., 2017) used unsupervised learning to overcome the limitations in the ground truth availability in disparity estimation. Thus, they carried their work using only the stereo images to infer the image pairs' disparity map. Their network of choice was the DispNet (Mayer, et al., 2016). However, the authors used supervised learning to finetune for Kitti dataset using the Confidence Guided loss function, which requires information about the original gourd truth to be estimated.

The implementation of networks that are dedicated to the disparity estimation witnessed much progress since the first implementation. (Liang, et al., 2018) suggested two steps for disparity prediction; the first step utilized the feature constancy correctness to estimate the initial disparity map, the second step uses the feature constancy for disparity refinement in deeper stages. The feature constancy borrowed from the earlier optical feature correction step stages is the layer after the data input layer. The feature constancy was obtained from the correlation of the features and the estimated reconstruction error. In the refitment stage, the initial disparity estimated, the correlation between the stages from the previous stage, and the reconstruction error are used to estimate the residual disparity, which eventually added to the initial disparity for a better disparity prediction.

The authors (Jammal, et al., 2017) estimated the disparity maps at different scales and incorporated these disparities to calculate the final disparity map. Using different disparity scales allows estimation for varying levels of disparity to be refined. For instance, course disparity is better to estimate close objects from the camera, while fine disparities are providing a better assessment of far objects that require detailed representation. Combining both cases allows a better estimation of the network for different network scales.

Yin & Shi, 2018 trained a networked coined GeoNet to estimate jointly monocular depth, optical flow, and camera motion from video. Their work is based on the 3D scene geometry that can be divided into two entities, first, static surfaces like roads and buildings, second, a dynamic object like cars and

pedestrians characterized by large displacement. Their network architecture consists of two stages to solve depth estimation and camera pose estimation, followed by optical flow estimation.

It could be more beneficial to use one image to reconstruct the 3D representation of the image. Humans can obtain enough information about the depths of the objects within an image using different cues. For instance, (Saxena, et al., 2007) used local features to represent image depth following the well-known computer graphics 3D model representation. Their proposal includes the over-segment images into patches of small coherent regions where all the pixels have similar properties. These small batches can represent small local planes that can infer the location and orientation of image patches during the test phase. To overcome the locality issue, they incorporated MRF to join the surrounding local features for better depth estimation. However, the method lacking the global context within the image and has poor representation for thin structures.

Another local approach (Ladicky, et al., 2014) utilizes semantics in the model operation, which allows a better understanding of the pixel depth estimation. (Karsch, et al., 2014) attempted to provide consistent image predications by copying whole depth images from the training set during the test phase. The method disadvantage is that it requires the entire training set to be available during the testing, which is not a practical approach.

Unlike previous studies, (Eigen, et al., 2014) produced dense pixel depth by utilizing two scales deep network. The training performed on images and their corresponding depth values using raw pixels values to learn features. Other studies built on the aforementioned work used Conditional Random Field (CRF) to increase accuracy (Li, et al., 2015) and modify the loss function from regression to classification or other robust loss functions (Cao, et al., 2017).

Using CNN, (Zbontar & LeCun, 2016), employed a Siamese network to estimate matching distances between patches obtained from images. Zhang, et al. used cross-based cost aggregation to estimate disparity by the same concept. Furthermore, the authors (Hirschmuller, 2005) used semi-global matching (SGM). However, the above approaches didn't train the networking end-to-end.

(Godard, et al., 2017) implemented DisNetS with some modification to estimate scene depth. During training, the network consists of two channels to estimate the left and the right disparities. Instead of depending only on the photometric loss function to estimate the error during the estimation, which provides low-quality disparities, they incorporate a disparity consistency checking to improve their results. During testing, though, only one image is required to estimate the depth of the scene.

In (Mayer, et al., 2016), the authors implemented a disparity estimation using a convolutional neural network with supervised learning. They built a synthetic dataset, coined FlyingThings3D, to facilitate the network training process. Two FlowNet network structures, initially proposed in (Dosovitskiy, et al., 2015), were adopted. The first one is called DispNetS, which accepts stacked stereo image pairs. In contrast, the second, called DispNetC (or DispNetCorr), initially use two branches that meet at a correlation layer forming a single fused encoding branch. The correlation layer is used to aid matching features from two input images. The networks implement multiscale reconstruction (decoding) with explicit supervised loss at reconstruction levels contributing to the overall loss function being minimized during the training process. This arrangement allows the training to be performed at each stage to improve network training. The authors first developed the FlyingThings3D synthetic dataset to help in the training phase and then finetuned with other datasets like Kitti (Geiger, et al., 2013) and other datasets.

In this work, a DispNetC architecture is adopted with modification to compute the disparity of the image pair. Two of the DispNetC with shared parameters are used. This joint feature allows simultaneous and synchronized training of the two networks to produce compatible features representing the forward and backward disparity. Having consistent estimation for the forward and the backward disparities allow assessment for the occlusion mask calculated in the forward using the backward disparity and the backward occlusion mask using the forward disparity. Thus, the network sharing the necessary information regarding the occlusion area. Having access to the occlusion information allows better error computation during loss estimation and reduces the wrong estimation effect when the occlusion areas are present.

### 5.2.5 Disparity Estimation Challenging Tasks

Generally, algorithms that use computer vision techniques to estimate disparity are based on estimating the horizontal displacement of an object between two images provided they are rectified. Most of the previous work in disparity uses computer vision techniques with very small datasets and developed with controlled lighting conditions. These datasets did not provide real-world cases like repetitive patterns, reflective surfaces, textureless areas, and thin structures that can be seen in a natural scene. While previous computer vision algorithms estimate disparity from local features, they couldn’t comprehend the objects' global relation within the scene, which was considered challenging, especially with the natural scenes dataset like Kitti.

In contrast to the local processing nature algorithms, DL methods are learning simultaneously powerful local and global representations directly from data in object detection tasks (Krizhevsky, et

al., 2012). Furthermore, DL models understand semantics (Huang, et al., 2016). These tasks are succeeded with supervised learning if the dataset is large enough. In this case, the model should learn object geometry as part of the prediction process. For instance, the model should understand the car that contains many transparent (windows) and surfaces light reflection as one entity. The model should rely mainly on the car's expected shape and compensate for smooth surfaces, transparent surfaces, and reflective surfaces as part of the object. For these cases, the model should apply consistent disparity correspondences that are local to that part. This understanding of the geometry is obtained by the contextual concept of the image sample's global semantic.

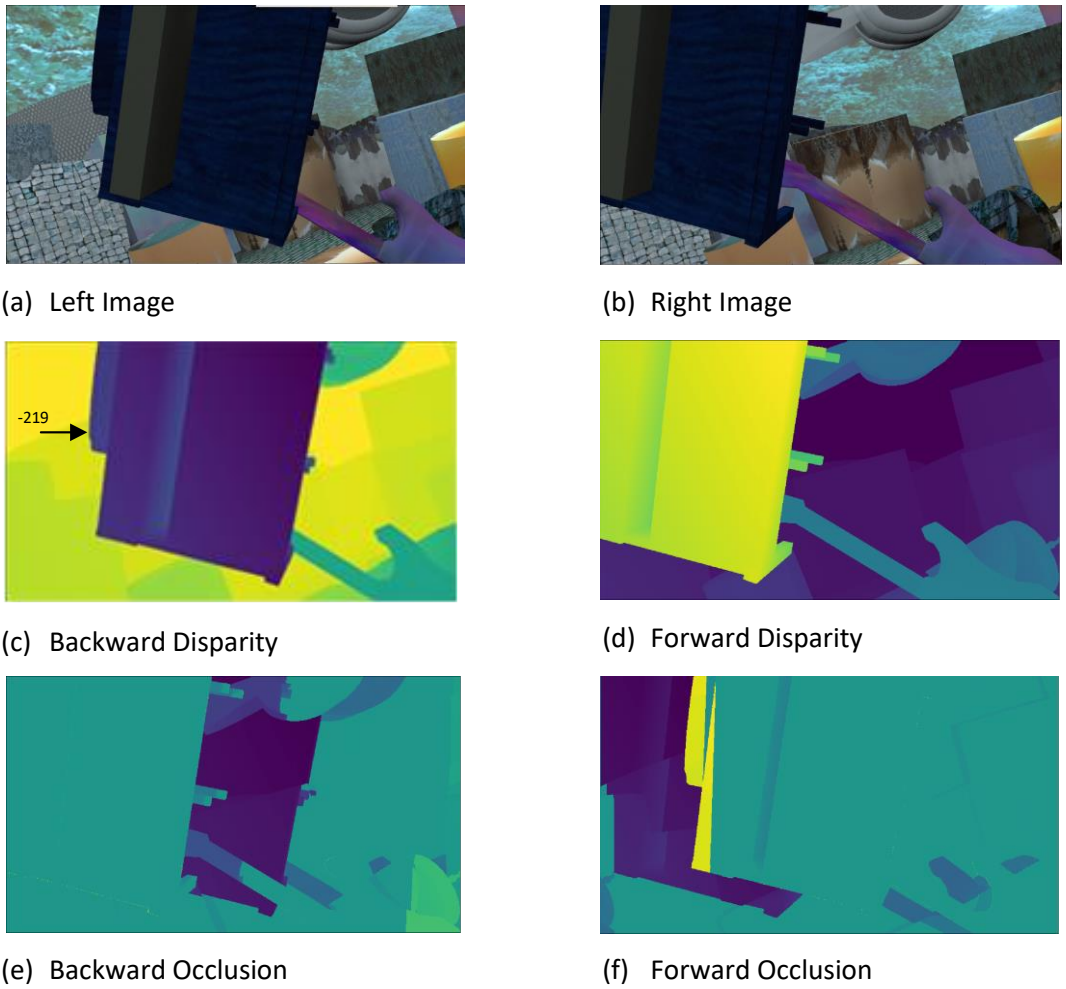
Hence, the model should be able to understand the global context to infer the local context. For instance, the representation of the reflected light area from the car body is a maximum value on the image channels (RGB). However, since the model understands the shape of the car, the expectation is that the model should smooth the spot and provide a disparity value close to the disparity prediction in the neighbourhood. This should also be applied to the light reflection or the transparent areas of the buildings.

For cases like the sky, where the pixels points are conceptually at infinity, the global semantics should be developed by the sky's local position, which is usually at the top of the image. Furthermore, the sky colour range is typically bounded by a high photometric discontinuity from other objects like buildings and trees. This discontinuity is another clue for the model to bound the sky area. In this case, the model should infer the zero value of disparity at these pixels.

## 5.3 Occlusion Effect on Disparity Estimation

The occlusion problem in stereo vision has been discussed previously in Chapter 2. The problem introduces confusion in multiple views setting because each camera will experience the occlusion from a different perspective. In an optimization process, the occlusion effect appears on the loss function estimate, especially in unsupervised learning. The occlusion will introduce ambiguity to the loss estimation that will eventually affect the optimization process and the model's training. In some cases, the occlusion ambiguity effect exceeds the brightness constancy loss value, which can affect the learning process of a deep model. The optimization process will interpret this ambiguity as a legitimate entry, and a new weight update will be propagated through the model to correct what is supposed to be wrong. The frequent occurrence of cases like this eventually leads to an optimization that does not reflect the actual error estimation process and will degrade the training outcome. Figure 5.1 shows the stereo image with their correspondence backward and forward disparities and the occlusions related. As it shows, the front object is close to the camera, which makes the disparity value large on

the pixel where the object is located compared to the objects and the background behind it, which has lower disparity values.




(a) Left Image (b) Right Image

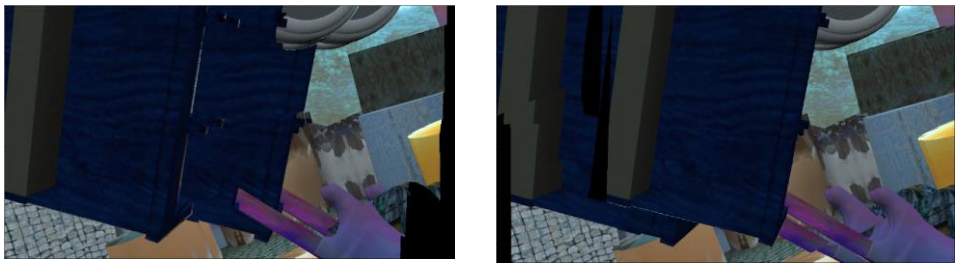
(c) Backward Disparity (d) Forward Disparity

(e) Backward Occlusion (f) Forward Occlusion

Figure 5.1: FlyingThings3D Dataset Samples. Stereo images with their corresponding forward and backward disparities and their related occlusion estimation. It can be seen that using the backward disparity to warp the left image to the right image introduces an unknown area of loss estimation. The image size is 560x960 pixels, and the green color in the occlusions images in (e) and (f) represents zero value, which means a non-occluded area. Other colors represent areas where the occlusion will deteriorate the learning process.

One of the methods that allow unsupervised learning is based on loss functions that are mainly depending on the reconstruction of the subject or moving image as close as possible to the reference image. The right image can be warped using backward disparity to compare it to the left image or visa-versa. For instance, using to warp the right image of Figure 5.1 to the left image with the forward disparity Figure 5.1(d) will introduce an ambiguity shown in Figure 5.1(f). All the areas that are not green colored are occlusion areas. These unknown values areas are subject to incorrect error values and outliers that will propagate back during the weights' updates.

Similarly, warping the left image to the right by the backward disparity results in a similar occlusion ambiguity effect. To give an example of the error introduced during the warping, Figure 5.2 is shown.



(a) Right image warped by forward disparity. (b) Left image warped b backward disparity.

Figure 5.2: Using the forward and the backward disparity to map the moving images toward the reference images.

In Figure 5.2(a), the right image has been warped to the left, as shown. The loss function compares the warped right image to the left image. The error values obtained for this comparison will not be accurate due to the distorted object's replica. As the object came closer to the camera, this ambiguity can increase substantially.

The warping process is an essential step to estimate the error between the reference image and the moving image. Therefore, the occlusion effect needs to be addressed by compensating for this error or avoiding the error estimation in the occlusion areas. As it has been shown, the occlusion area can be identified by warping the forward disparity by the backward disparity, or vice-versa. If the occlusion area is specified, the loss function can be masked by this area, and the effect of the occlusion ambiguity can be avoided.

## 5.4 Two Channels Disparity Estimation for Disparity Feature Consistency Checking

One of the techniques used in estimating and avoiding the occlusion is called disparity consistency checking (Hirschmuller, 2006), which is a task of assessing both stereo images' disparity. Each disparity image contains a different view of the scene and the occlusions, which can be exploited to estimate the occlusions across the scene area. The details about the disparity consistency checking with the estimation methodology and its implementation are explained below.

### 5.4.1 The Methodology

The method suggests dividing the pixels of the image into two types, the pixels with correct and clear correspondence between the two images and the pixels that are mainly related to occlusion. The

pixels with accurate correspondence will be utilized to estimate the loss function necessary to optimize a model, and the occluded pixels will not be considered during the loss calculation and will not contribute to the loss function. A binary mask will be generated that assigns a value of {1} for areas without occlusion and {0} for the occluded regions.

Mapping each image with the disparity of the other image will produce the pairs of Figure *5.2*(a, b). During the implementation of the training algorithm, the brightness constancy loss function depends on the version of these manipulated images to estimate the reconstruction quality. Now it can be seen that the object shape is corrupted in the occlusion area. The DL model will be self-supervised or guided by a metric of how well the images are reconstructed smoothly and accurately by the disparity map generated by the DL model. The warping process will be performed by incorporating part of the spatial transformer network module before evaluating the quality of the warping operation.

It can be seen in Figure 5.2 that the replica of the object areas located where the occlusion is occurring and it appears as a copy of the object overlayed on the original object. Therefore, implementing image registration with error in estimation will inherit the occlusion error and degrade the training process.

### 5.4.2 Utilizing the Forward-Backward Disparities in Occlusion Mask Estimation

The proposed method consists of estimating both the forward and the backward disparity and perform disparity consistency checking to generate an occlusion mask, as shown in Figure 5.1(e,f). The availability of forwarding and backward disparity allows generating the occlusion mask that the loss function acquires while estimating the intensity or brightness constancy constraint loss. In this way, the loss function will only consider the error obtained from pixels with a valid correspondence between the images, and pixels without correspondence will be ignored. Effectively, the occlusion mask will not appear in the same area continuously. Thus, it will not introduce bias to a specific area because of the object's arbitrary appearance within the scene.

The main relationship that can accomplish occlusion mask estimation is based on the backward disparity's warping with the forward disparity to estimate the backward occlusion (Sundaram, et al., 2010). Similarly, the forward occlusion mask can be calculated by warping the forward disparity with the backward disparity. The inequality below (Equations (5.6-7)) shows the relationship that extracts a mask of the occlusion area from the forward and the backward disparities.

$$\left|d^f(x,y) + d^b(x + d^f(x,y), y)\right|^2 < \alpha_1 \left(\left|d^f(x,y)\right|^2 + \left|d^b(x, d^f(x,y), y)\right|^2\right) + \alpha_2 \quad (5.7)$$

$$\left|d^b(x,y) + d^f(x + d^b(x,y), y)\right|^2 < \alpha_1 \left(\left|d^b(x,y)\right|^2 + \left|d^f(x, d^b(x,y), y)\right|^2\right) + \alpha_2 \quad (5.8)$$

Here, $d^f(x,y)$ and $d^b(x,y)$ are the forward and the backward disparities, respectively. The values of $\alpha_1$ and $\alpha_2$ control the tolerance of the level of precision between the level of the disparities, and they are of 0.01 and 0.5, respectively.

### 5.4.3 Network Implementation

The network below is inspired by the results reported in (Meister, et al., 2018). The authors proposed a novel network architecture called UnFlow, for unsupervised optical flow estimation from a pair of images. Here, a similar overall architecture has been implemented with modifications of the contractor-expand network originally proposed for estimation of disparity from stereo images (Mayer, et al., 2016). Figure 5.3 shows the detailed CNN used in the training and inferencing (without the reconstruction or loss function details).

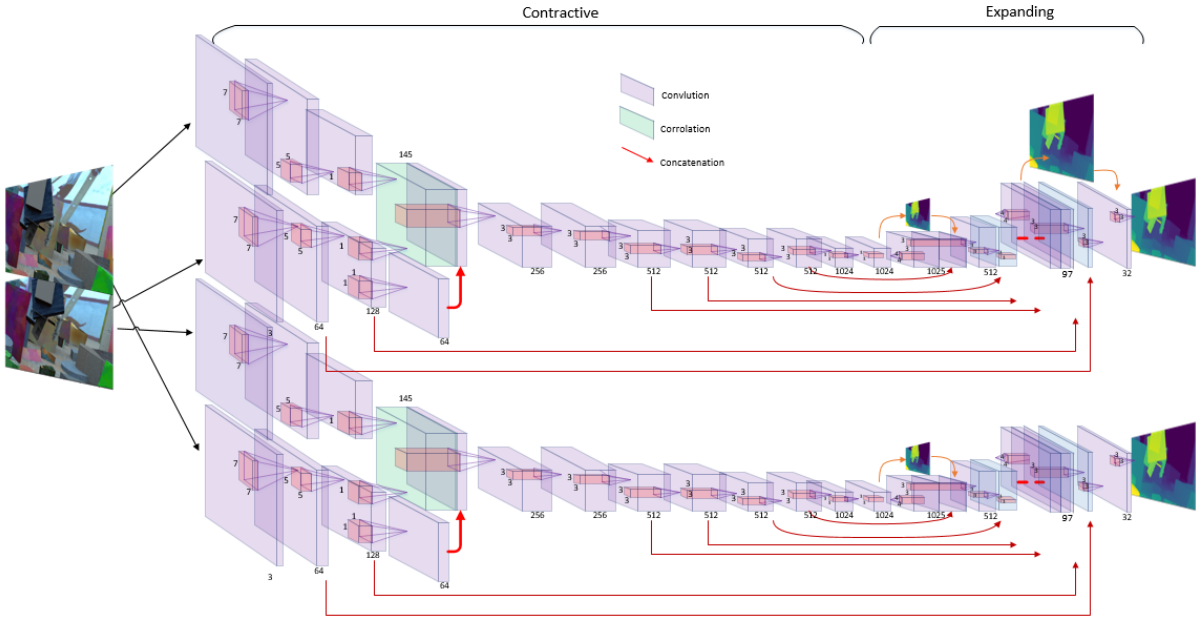



Figure 5.3: Proposed disparity estimation CNN model. The network reported in (Mayer, et al., 2016) (Odena, et al., 2016) has been duplicated to perform parallel channels. One channel produces the backward disparity feature while the other channel predicts the forward disparity features. Each layer of these channels has been set with weight sharing accepts the correlation layer since it contains no weights.

The modifications can be listed as follows:

1. Instead of the optical flow-oriented network, the DispNetC network has been used in the implementation.

2. The deconvolutional layers of the DispNetC had been replaced by an up-sampling layer followed by the convolutional layer. This arrangement provides more expressive features than the deconvolutional layer (Odena, et al., 2016; Girshick, 2015).

In this model, the first channel is fed with the two images in the forward order (say left-right), whereas the second channel takes input images in the reversed order (i.e. right-left). The model weights are shared across the two channels, allowing better feature learning because each task influences the other (Girshick, 2015) since the model simultaneously learning the forward and the backward disparities. Furthermore, weight sharing encourages parameter regularization while decreasing the model parameters size and reducing overfitting (Ullrich, et al., 2017). Moreover, weight regularization has also been applied to minimize overfitting further. But the main reason for the weight sharing in this implementation is to allow the model to respond precisely in the same way (i.e., with same logit levels) to the same feature as it progresses from the previous stages. The inference layer has no activation function and having weight sharing is an essential requirement to generate closely related inferred disparities (left and right). Closely related disparity values allow better estimation of the occlusion mask.

With two disparity branches, the network can be trained for forward and backward disparity estimation in a supervised fashion if a relevant ground truth is available. However, having a model trained with supervised learning produces extremely bad results if tested on another dataset, especially if no ground truth is available to fine-tune the model (Samer, et al., 2017). Concerning the disparity estimation, the available dataset with ground truth is either synthetics as in FlyingThings3D, which contains the computer-generated scene with their related foreground and the background disparity ground truth. Or, the ground truth can be generated from LIDAR (Light Detection and Ranging) point cloud data and has been projected to produce the disparity ground truth as in the case of Kitti dataset. Therefore, it is entirely reasonable to train a model in an unsupervised setting as it will liberate the learning from the requirement of ground truth. It will allow for a further fine-tune to other datasets without ground truth.

In this work, the model trained in an unsupervised fashion needed a mechanism to estimate and evaluate the model's performance by equipping it with a suitable image reconstruction module. To achieve the evaluation, part of the network like the Spatial Transformer Network (STN) (Chapter 3) has been employed. This model consists of a spatial transformation network model, and it performs the transformation obtained from the network inference. Here, two parts are involving in reconstructing or warping the image, the grid generator, and the interpolator function. Thus, the

disparity map containing the pixel's mapping in the x-direction is summed to a grid with the same dimension (batch size, height, width, channel). The resulting grid is utilized to interpolate the image.

Figure 5.4 shows the warping module that appears on each stage of the expander network. As it shows, the reconstruction or the warping of the image will be performed at each scale. For an image of (384) height and (768) width, the first reconstruction will be performed at the size of (24x48) pixel, and it duplicated at each stage.

Also, it can be seen that the input images themselves are resampled at different image scales with a power of 2 and for five levels that fit the scale of the image mapping parameter (disparity map), which is a requirement for the reconstruction process.

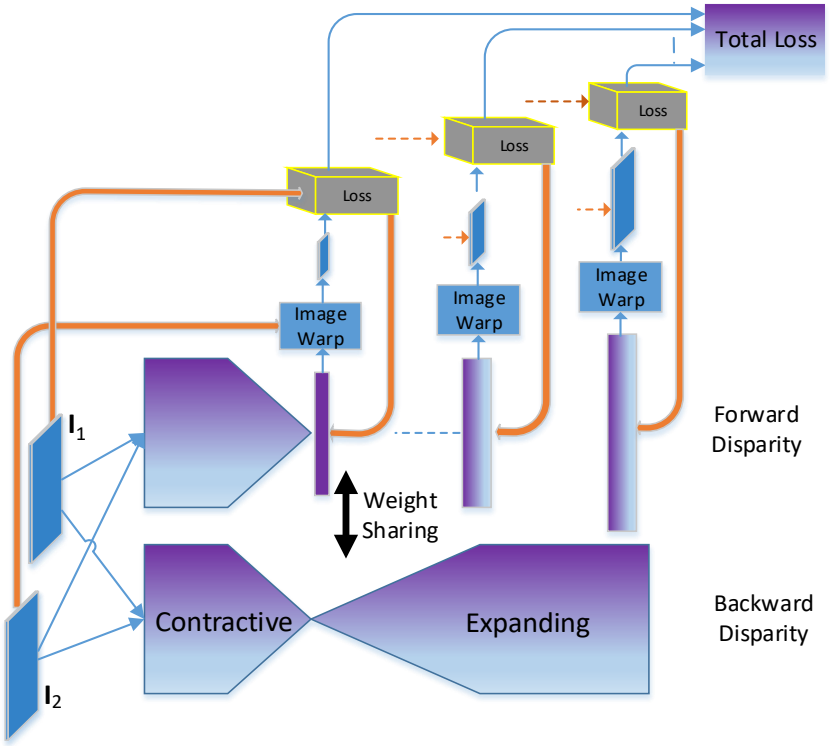



Figure 5.4: The details of the warping module in the proposed network. The same arrangement that is shown on the upper side of the network is also available on the lower side of the network, and it is not shown.

## 5.5 Occlusion Aware Loss function and other smoothing loss functions

With regards to the loss functions that have been implemented for this model, at each expanding part's reconstruction level of the model, the downscaled input images for both the left and the right images are provided. In each scale and each channel, the loss function estimating how good is the reconstruction or fit the warping of one image with respect to the other after mapping it by the corresponding disparity map. In this case, the loss function will evaluate how each pixel has been mapped to its corresponding pixel in the other image. This evaluation type is based on the brightness

constraint explained in (Chapter 2). The evaluation of mapping is performed using the robust generalized Charbonnier loss function:

$$g = (x^2 + \epsilon^2)^{\alpha} \tag{5.9}$$

where $x$ is the photometric difference between the image and its warped version. $\epsilon$ is a small positive value of $10^{-6}$ and $\alpha \in \mathbb{R}$ (Barron, 2019). Charbonnier's loss function behaves like L2 loss function near the origin and like L1; otherwise, researchers sometimes refer to it as L1-L2 loss function. Charbonnier loss function generally robust as its low sensitivity to outliers. This behavior stabilizes the learning process and provides bounded gradients that contribute to the disparity map's smoothness. The loss function uses estimated displacements to maximize the photometric consistency (minimize the data fidelity term) between corresponding warped images.

However, in this work, the Charbonnier loss function has been modified by introducing the occlusion mask on the estimated error, Equation (5.10):

$$\begin{aligned} E_D = \sum_{(x,y)\in P} & m^f(x,y).\, g\left(I_1(x,y) - I_2\big(x + d^f(x,y), y\big)\right) \\ & + m^b(x,y).\, g\left(I_1\big(x + d^b(x,y), y\big) - I_2(x,y)\right) + \beta_1(1 \\ & - m^f(x,y)) + \beta_2(1 - m^b(x,y)) \end{aligned} \tag{5.10}$$

where, $\beta_1 = \beta_2 = 0.1$. The function $g$ is multiplied by the mask function that effectively nullifies the values where the occlusion occurs. As the error function needs to be minimized, the first terms will be easily minimized by increasing the occlusion area (i.e., all the mask values to be zeroed), which is not the aim. The third and the fourth terms are added to control the occlusion area not to exceed the estimated area. It is also important to mention that the mask is calculated by comparing pixel to pixel values between the two disparity maps. The inequality in Equation (5.6,7) will produce null values for pixels with the same disparity values. These masked areas will enhance the disparity estimation because the error is focusing only on enhancing the correct values, and it corrects the mistakenly predicted values in the subsequent training iteration.

For the smoothness constraint part of the disparity estimation, different loss functions have been implemented for this network to address different smoothing cases. For example, low gradient areas (i.e., smooth and monotonic object's surface or motion with slow transition across the image plane) are considered problematic issues (Chapter 2). Cases like these require filling or smoothness capability

covering the area with pixels of the same values as they logically should belong to the same object. When the object possesses texture, smoothness can be easily obtained by minimizing the gradient of the disparity. This operation effectively reduces the differences between the adjacent pixel in the disparity map. This action is legitimate because although the object has changed texture, it moves at the same speed, which means that the disparity pixels related to the object should be equal, Equation (5.10).

$$Smooth\_loss1 = \sum_{x,y=R} (d_x) + \sum_{x,y=R} (d_y) \tag{5.11}$$

Another loss function that has been utilized for the smoothing matter is focusing on edge smoothing. Usually, sharp edges are easy to be learned by the model. It will appear on the disparity map compared to a smooth surface, which has very low features. This tendency emphasizes the edges such that it will appear as high values compared to the surface connected to it, which introduces error in the estimation. To cope with that, a loss function that considers the edges has explicitly been incorporated, Equation (5.12).

$$\begin{aligned} E_s = \sum_{(x,y)\in P} & \left|d_x^f(x,y)\right| . \exp\left(-|I_{2,x}(x,y)|\right) + \left|d_y^f(x,y)\right| . exp\left(-|I_{2,y}(x,y)|\right) \\ & + \left|d_x^b(x,y)\right| . exp\left(-|I_{1,x}(x,y)|\right) + \left|d_y^b(x,y)\right| . exp\left(-|I_{1,y}(x,y)|\right) \end{aligned} \tag{5.12}$$

where: $A_x = dA/dx$ and $A_y = dA/dy$ are the image derivatives. Firstly, the images' gradient is generated in the x and y directions for both the left and the right images. Then, a weighting function introduced to suppress high values of gradients and regulate the estimation. Then, the weighted value obtained is multiplied by the gradient of the disparity. This arrangement allows the image to contribute to the weighting of the disparity regulations. The gradient will be high for a sharp image edge, so it is suppressed by an exponential function and multiplied by the disparity gradient. This arrangement allows a weighted value for this case. Similar, for a less sharp edge, the weight will be higher, and the returned disparity gradient will not be drastically different in weight to the case when the edge is sharp. For that, the resulting gradient image that is mainly addressing the edges of the image is smoothed.

In addition to the above loss functions utilized in estimating the total losses, another loss function is employed. This function is an intermediate result required to obtain the occlusion mask. Figure 5.5 shows a sample of this loss function. This function is targeting the inconsistency in the estimation of the disparities between the left and the right disparities in addition to the areas where the occlusion appears. It can be seen in the figure that the model is experiencing a problem when estimating

disparity for far distance objects, which is a well-known issue due to the non-linearity in the estimation (Figure 2.10). This type of measure is considered as uncertainty in the disparity estimation, and this function contributes to the minimization of this uncertainty problem, and it will be investigated later in this chapter.

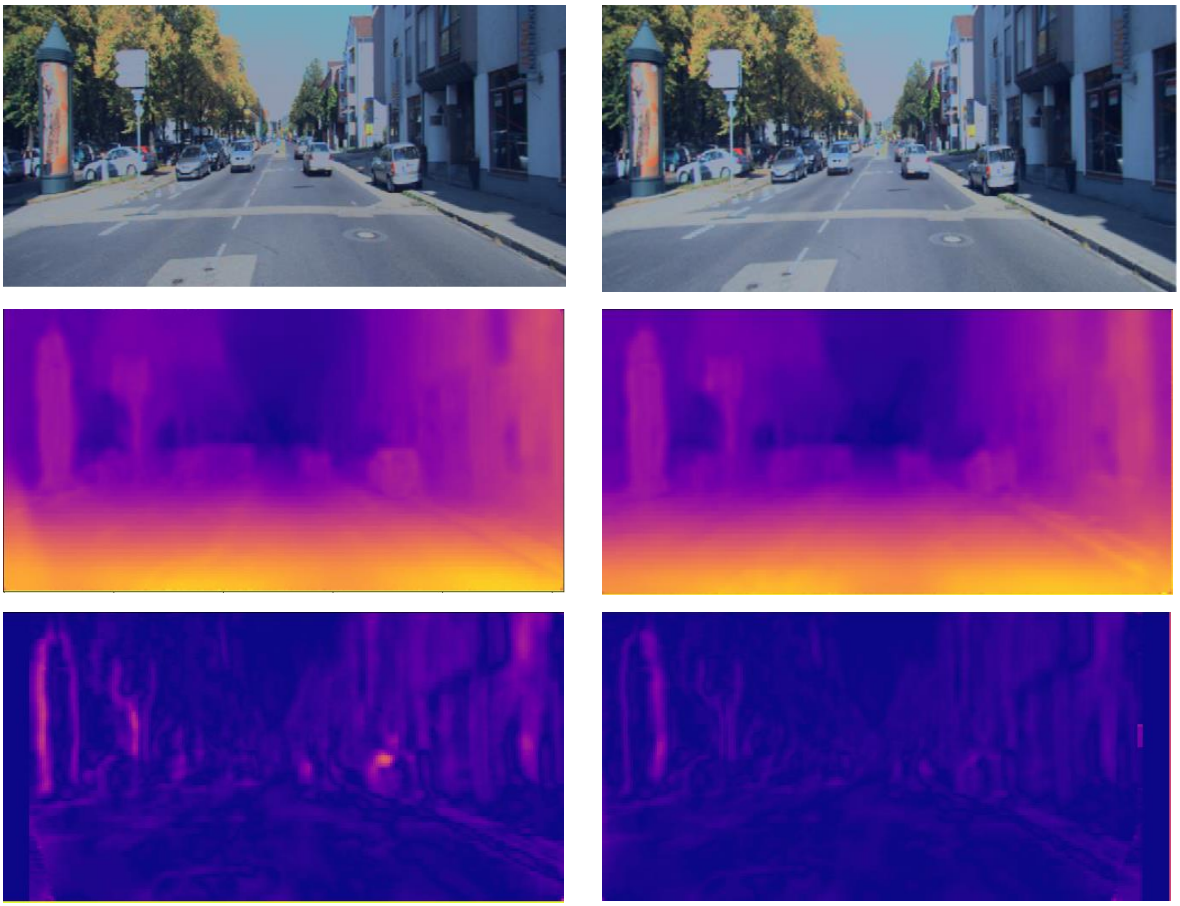


Figure 5.5: Consistency Feature Map. Top: stereo image pair, Middle: Forward and Backward disparities, Bottom: Consistency Feature Map shows the area where the occlusion appears as a hot area.

The loss function related to the consistency feature that has been used in this work can be expressed as:

$$E_c = \sum_{(x,y)\in P} \left|c^f(x,y)\right| + \sum_{(x,y)\in P} \left|c^b(x,y)\right| \tag{5.13}$$

## 5.6 Datasets and Data Augmentation

For training and evaluation of the proposed network, three publicly available datasets that are well-known by the research in this field have been utilized. The dataset included FlyingThings3D (Mayer, et al., 2016), KITTI (Geiger, et al., 2013; Menze & Geiger, 2015), and CityScapes (Cordts, et al., 2016). Each

one of these datasets produced with different settings and addressed a specific purpose. Settings like stereo baseline length and camera focal length are part of these settings that affecting the scene perspective properties and the correspondence relation between stereo images. The deep learning model will learn these characteristics and the dataset settings once it is trained with them. To overcome the differences between the datasets settings, a finetuning is necessary in some cases, mainly to adjust the DL model for the new settings and their related prospective properties.

Before training or testing a DL model, the training image pair need to be calibrated against camera distortion (removing the non-linear distortion due to the camera lenses and other non-linear effects). Image calibration helps avoid the situation that a DL model learns these distorted properties and affects the results if an image from a different camera source is tested. Also, the images need to be rectified specifically in the case of stereo based disparity estimation (i.e., each row of pixels in the left image directly corresponds to the same row in the right image).

Regarding FlyingThings3D dataset, a synthetic dataset of image size 540x960 pixels contains small objects that are moving around a stereo camera in a virtual environment. It consists of a large detailed synthetic dataset specifically built to support disparity, optical flow, and scene flow estimation. The virtual imaging sensor has 32.0mm x 18.0mm, with a virtual focal length of 35mm. The image pairs are undistorted and rectified as required and presented in two versions, *cleanpass* and *finalpass datasets*. The *cleanpass* setting includes lighting and shading effects but no additional effects. In contrast, *finalpass* images also contain motion blur and defocus blur. The FlyingThings3D stereo RGB images are renderings of scenes consisting of different randomly distributed 3D objects. Generally, the background of the scene consists of cylinders and cuboids that vary in scale, texture, and orientation. The foreground objects consist of 37927 detailed 3D models from Stanford's ShapeNet dataset (Savva, et al., 2015). Between five to twenty objects are randomly sampled from these foreground object, textured, resized and rotated along a smooth 3D trajectories with random displacement. The resulting rendered images are available as clean (cleanpass) or as more realistic (finalpass) versions. The latter include motion and depth of field blur effects. In this work, round 22000 stereo image for training and 4000 for both validation and testing were employed from this dataset. The corresponding forward and backward ground truth disparity data are also available. The purpose of this dataset is to train a deep learning model to work for internal scenes that required estimation for optical flow or disparity, or it could be a staging step to train a model than finetune the model with other datasets like Kitti.

Kitti dataset is a challenging real-world computer vision benchmark. The dataset provides a small amount of ground truth for stereo, optical flow, visual odometry, 3D object detection, and 3D tracking

for evaluation purposes. It has been gathered by equipping a standard station wagon with two high-resolution color and grayscale video cameras. The ground truth was obtained by a roof-mounted Velodyne laser scanner and a GPS localization system. The scenes are captured by driving in urban areas of Karlsruhe's mid-size city, in rural areas, and on highways. The scene may contain cars and pedestrians, traffic lights, cyclists, and many other details on average. The aim of developing the Kitti dataset is to cover the low-performance gap on models that could be trained on a small dataset like Middlebury (Scharstein, et al., 2014). Moreover, it covers the low-performing model when tested for real-world environments by providing a large amount of data that fits deep learning training requirements with novel difficulties to the computer vision community.

For Kitti datasets, the intrinsic and the extrinsic (for the camera and LIDAR and between the stereo camera) parameters are provided in calibration files. However, the provided dataset is calibrated and rectified, and it can be used for DL model training and testing without further processing. The KITTI disparity dataset consists of 2012 and 2015 versions. The “KITTI Stereo 2012” consists of about 50K undistorted and rectified image pairs of 375x1242 pixels that were collected to develop mobile robotics and autonomous driving applications. The images are collections of about 6 hours of real traffic recordings. Moreover, the “KITTI Stereo 2015” set incorporates 200 image pairs with corresponding disparity, semantic segmentation, and other gourd truth. In particular, the disparity/depth ground truth is collected from a point cloud data obtained from the attached LIDAR.

CityScapes is another dataset that is designed to capture street scenes with high variability (Cordts, et al., 2016). The dataset was collected for several months during different seasons and weather in 50 cities. The RGB camera pair had a 22 cm baseline with 2MP sensors and operated at 17 frames per second. The cameras were calibrated, and the image pairs were rectified. The dataset consists of around 20000 images pairs of 1024x2048 pixels with forward disparity ground truth provided. In addition to the rectified image pairs and their corresponding data, the dataset includes vehicle odometry data obtained from vehicle sensors and GPS tracks. CityScapes dataset focuses on semantic understanding of urban street scenes. The dataset contains a 20k image of a weakly annotated image and 5k of finely annotated images. It provides annotation for semantic, instance-wise, and dense pixel annotations with 30 classes for semantics segmentation.

Each dataset has its limitation, FlyingThings3D is synthetic, and a model trained on this dataset will need further training to adapt it to a real-world scene. Furthermore, perform supervised training on this dataset will require supervised fine-tuning, which is a limitation. In the case of the Kitti dataset, the dataset addresses the challenging real-world scenes, and the fact that there is no ground truth

available for this dataset is a limitation. Cityscapes also have the limitation of not being completely annotated because the major part is weakly annotated, which makes it not entirely useful for supervised training, especially for pixel-level prediction. In addition to that, the actual images are heavily processed, and it is an almost syntactic dataset, which makes it less useful for real-world use, as in the case of the Kitti dataset. From the above, the choice of the work is to implement a deep learning model that can perform a self-supervised or unsupervised learning.

Data augmentation is an essential step in the training phase of the network. It discourages the network from overfitting the dataset since the data queue will receive new data at each data sample acquisition. Each sample pair passes through random cropping from the original data and then passes image property augmentation, including gamma uniform random manipulation with a range between [0.8 - 1.2], random brightness manipulation in the range between 10%-50% from the minimum value of the image brightness. The contrast also uniformly changed in the range between [0.5-2]. Moreover, per channel contrast is performed in the range between [0.8-1.2]. Then a saturation clipping is performed to adjust the image level between [0-1].

## 5.7 Unsupervised Training with Warping Images

In supervised learning, the ground truth is usually available to perform the model's necessary training by comparing the model results to the provided ground truth. On the other hand, unsupervised learning requires a different way of error estimation (Bengio, 2012). The error in the model output could be estimated by comparing the original image to the output reconstructed image, as in autoencoder training. In the case of the disparity and optical flow estimation, the error is estimated by comparing the image to its reconstructed version. The Euclidian distance between the image and its reconstructed or warped version can be considered an error metric to measure the network performance.

### 5.7.1 Training Procedure

The expanding part of the network starts with a small scale of 12*24 up to the final scale of 192x384. The training was performed from the smallest to the largest scale in steps. Each training step consists of a 10k iteration. Thus, the backpropagation is starting from that specified scale back to the network bottom (the input). On the next 10k iteration, the next higher scale will be trained back to the network bottom. As the training schedule moves to the next scale, the weight of the loss related to the previous scale will be reduced by around 60%. In the final set, the stage schedule zeroes the losses from the smaller scales and accept only the final layer where all the loss will be calculated with respect to that

last layer. Within this loss schedule weighting scheme, the network will be trained gradually from the earlier stage in the expanding layer to the top of the last stage. The network will reconstruct the same features at each scale with double the size of the previous layer. This technique allows for better training. In principle, the large or big object disparity will first appear in the network's lower layers of the expander. It means that the object will have early correspondence information in the early stages of the DL model. The object size will be duplicated in the next stage as the feature size will be upscaled. A big object in the scene represents a large disparity value. Having an object disparity representation upscaled gradually is a reliable method to obtain stable and repeatable results. This approach mimics the approach of the Laplace scaling technique.

The disparity map appears based on the loss function estimation between the warped image and the reference image. The optimization of the model is performed based on the capability of the model to produce the best mapping (or disparity) to fit the moving image to the reference image. As the model starts to produce a better disparity map, the mapping's quality will improve, and the difference of the error between the two images will reduce to its minimum. Moreover, the smoothing loss function will encourage the network to fill and trim the generated disparity and close the gaps that may appear due to smooth surfaces or ripples due to surface textures.

During the training, the network estimates the occlusion mask by feeding the forward and the backward disparities to the occlusion mask module. In this model, Inequality (5.6-7) will be performed, and the resulting masks will be generated for both the forward and the backward disparities. For example, Figure 5.5 shows the estimation of the occlusion mask that will be produced with regard to the sample image pairs. The mask consists of binary values; binary {0} is where the mask is waiving the occlusion areas. The loss function will be estimated with respect to the binary {1} pixels in the mask image. Having the mask ignoring this area improves the training process; otherwise, the specified area will introduce error in estimation because the warped image will have zero values in these areas while the reference image will have pixels with values. The last group of black colored pixels is mainly related to the wrong disparity estimated values. This inaccurate estimation is primarily happening in areas where the object is far from the camera, or there is some transparency in the image.

Below is a detailed explanation of how the warping function developed.

### 5.7.2 Image warping module

The image warping module is part of the Spatial transformation network explained previously. The module contains the grid generator and the image sample modules. The grid generator will generate a grid with the dimension of the image in the corresponding scale. As explained in Chapter 3, the grid consists of two vectors x, and y. Each corresponding element in the two vectors represents the position of a pixel in the reconstructed image. In the disparity estimation case, the primary mapping will be performed on the grid's x-direction. The disparity map is added to the x-grid, which means that the pixel's position will contain the pixel's modified position in the moving image for this pixel will move to. The modified grid passes to the interpolation function to perform the sampling of the moving image based on the modified grid. Generally, the grid and the disparity map are scaled between [0 - 1], but the image height and width scale the modified grid values at the specified scale during the interpolation. The interpolation performed is based on the Bi-Linear interpolation. The resulting image should be as close as possible to the reference image. In the proposed network's case, the right image is mapped or warped towards the left image by the forward disparity, and the left image is mapped to the right image by the backward disparity.

### 5.7.3 Analysis and Results

For Disparity Estimation, there are mainly three measurement methods that estimate how the model performs. End Point Error (EPE) is a metric used in many literacies for this topic. Equation (5.14) represents the EPE, which is the same definition as L1 score.

$$EPE = \frac{1}{M * N} \sum_{(x,y) \in P} \left| d(x,y)_{prediction} - d(x,y)_{groudtruth} \right| \tag{5.14}$$

The second qualitative evaluation is based on the range of the error in pixels between the predicted and the ground truth disparity map. For >3 percentage error measurement, if the difference between predicted and ground truth is ( > 3 pixels), an accumulator will increase. The resultant accumulated value will be divided by the number of image pixels. This number represents how many in percent of pixels have a value bigger than 3 pixels. Similarly, the ( > 5-pixel) percentage represents the definition. For these three methods, as lower are these metrics as better results are obtained.

Table 1 shows the results obtained using the network architecture and the loss function described in the previous sections. Three metrics have been used for the proposed method evaluation. The End

Point Error (EPE) metric is an average absolute error measured between the estimated and the ground truth disparity maps, and the two other metrics measure the percentage of pixels with the disparity error bigger than 3 pixels (>3P) and 5 pixels (>5P). The proposed architecture was tested using different training/testing data and learning regimes, including supervised, unsupervised, and fine-tuning learning scenarios.

The results for five different experiments are reported in Table 1, with the results from each experiment provided in the table's successive rows. In the first experiment, the proposed network is trained in a supervised manner on the 22,000 synthetic image pairs selected from the FlyingThings3D dataset. The performance of the network is evaluated on different 4000 FlyingThings3D image pairs. The next two experiments use the pertained network from the first experiment and fine-tune the network using unsupervised training on the KITTI and CityScapes datasets. The second experiment uses 40,000 image pairs from the 2012 KITTI subset for training and 200 image pairs with known sparse ground truth from the 2015 KITTI subset for testing. The third test uses 20,000 CityScapes image pairs for training and different 1525 image pairs for testing. The fourth experiment uses the same data for training and testing as the first experiment. Still, the network is entirely trained in an unsupervised paradigm. i.e. without any supervised pre-training. The last experiment uses the same training and test KITTI images used in the second experiment, but again no supervised pre-training was used.

It can be concluded by comparing the results of the first and fourth experiments that supervised learning provides better results than unsupervised learning. This should be expected as supervised learning uses much richer information. Interestingly, when comparing results from the second and fifth experiments, a supervised pre-training using a synthetic dataset could somewhat improve results on the real data, even though the nature (properties) of the used synthetic data is very different from the real data. This is an important result as it is relatively easy to generate a large synthetics dataset with the ground truth, i.e., suitable for supervised training. In contrast, it is very difficult or even impossible to obtain ground truth disparity maps for some scenes in real data acquisition scenarios.

The results reported in the first line of Table 5.1 are better than the corresponding results reported in (Mayer, et al., 2016). The performance improvement could be explained due to the occlusion estimation and the shared parameters between the forward and the backward disparity networks. The loss function that the model trained with is the Mean Absolute Error (MAE) or L1 loss function that compares the predicted disparity with the ground truth. This network trained for maximum displacement for the correlation layer of 40 pixels as suggested since the required displacement in the

FlyingThings3D dataset is high compared to other datasets. In the case of KITTI model, the maximum displacement parameter was set to 20 pixels.

Some additional work is planned to investigate network generalisation properties with supervised, unsupervised, and fine-tuned scenarios. In that case, the network is to be trained (fine-tuned) and tested on data from different datasets, including indoor and outdoor cases. Note that the fine-tuning in this context is to train the network without ground truth.

| **Dataset** | **Training** | **EPE**[pixel] | **>3P** [%] | **>5P**[%] |
|---|---|---|---|---|
| FlyingThings3D | Supervised | 1.4 | 13.43 | 3.99 |
| FlyingThings3D fine-tuned on KITTI | Unsupervised | 2.05 | 9.70 | 2.14 |
| FlyingThings3D fine-tuned on CityScapes | Unsupervised | 1.96 | 23.84 | 6.61 |
| FlyingThings3D | Unsupervised | 2.50 | 25.2 | 12.24 |
| KITTI | Unsupervised | 2.02 | 9.81 | 2.42 |

Table 5.1: Values of different metrics obtained for different training/testing scenarios.

Qualitative results are shown in Figure 5.6-7. Figure 5.6 shows results obtained for KITTI dataset. In that case, only limited disparity ground truth is obtained from LIDAR is available. The predicted disparity map pixels are nullified, where the corresponding pixels in the ground truth were null. The proposed model can inference the disparity maps with 24ms (40 fps) on Nvidia Titan Xp and 60ms (16fps) on Nvidia GTX960, which showing real-time performance capability.

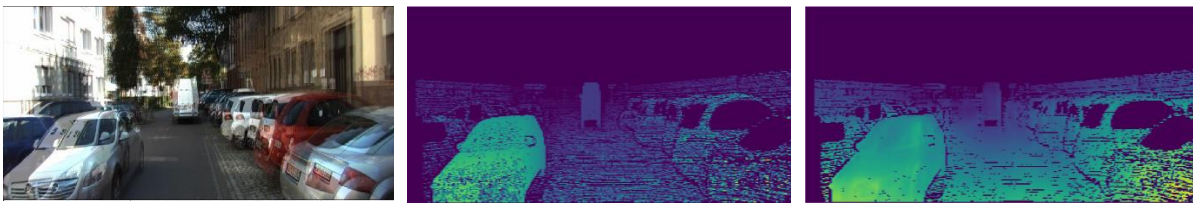


Figure 5.6: A sample result from the fifth experiment - the KITTI dataset. (Left): Input left-right images, (Middle): Kitti disparity ground truth, (Right): predicted disparity using unsupervised trained.

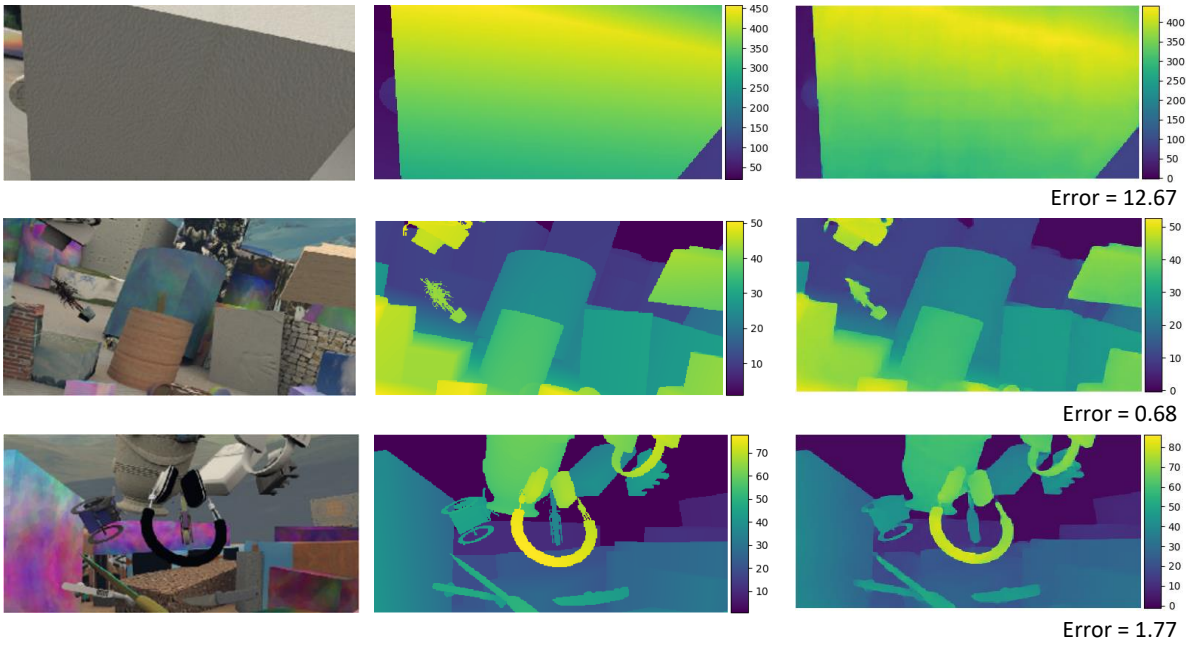



Figure 5.7: Disparity model qualitative test with FlyingThings3D. A sample result from the fourth experiment - the FlyingThings3D dataset. (Left): Left image, (Middle): The corresponding disparity maps ground truths, (Right): The estimated disparity maps using unsupervised training. Under each estimated disparity map, the error with respect to the GT is reported.

At this stage, the stereovision camera used in generating the FlyingThing3D dataset has a baseline of 1.0 feet, about 30.50cm, with a focal length of 35.00mm. The developed model has been tested for a different stereovision camera than the one used for the training dataset. A Grey Point Bumblebee 2 camera was utilized to obtain new images that is completely different from the FlyingThing3D dataset. This camera has a baseline of 12cm with a focal length of 3.8mm. Figure (5.8) shows a stereo image of an office where this study has been performed. In the figure, the left and the right images are provided; however, no ground truth is available. But the model output for this image can show how the model response to different image dimensionality that may be useful to discuss. It can be seen in the obtained disparity map that the quality of the output is reasonable, and the model can identify the proximity of the object from the camera. For instance, the shelf on the right is closer to the camera than the black chair on the back.

The estimated disparity, however, suffers two issues. First, knowing that the disparity/depth is directly related to the focal length and the baseline values, Equation (2.44), the model's obtained disparity estimation is not the best approximation to the expected disparity. However, the acquired disparity map may be correct, to some extent, using Equation (5.14):

$$D = D_{pred} \frac{f_{L_2} b_2}{f_{L_1} b_1} \tag{5.15}$$

where $D_{pred}$, $f_{L_1}$ and $b_1$ are the model disparity response, focal length, and baseline of the test camera (Grey Point Bumblebee 2), respectively. Both $f_{L_2}$ and $b_2$ are the focal length and the baseline

of the dataset, FlyingThings3D in this case. However, Equation (5.14) may not be generalized to the pixels on the left and right edges of the image.

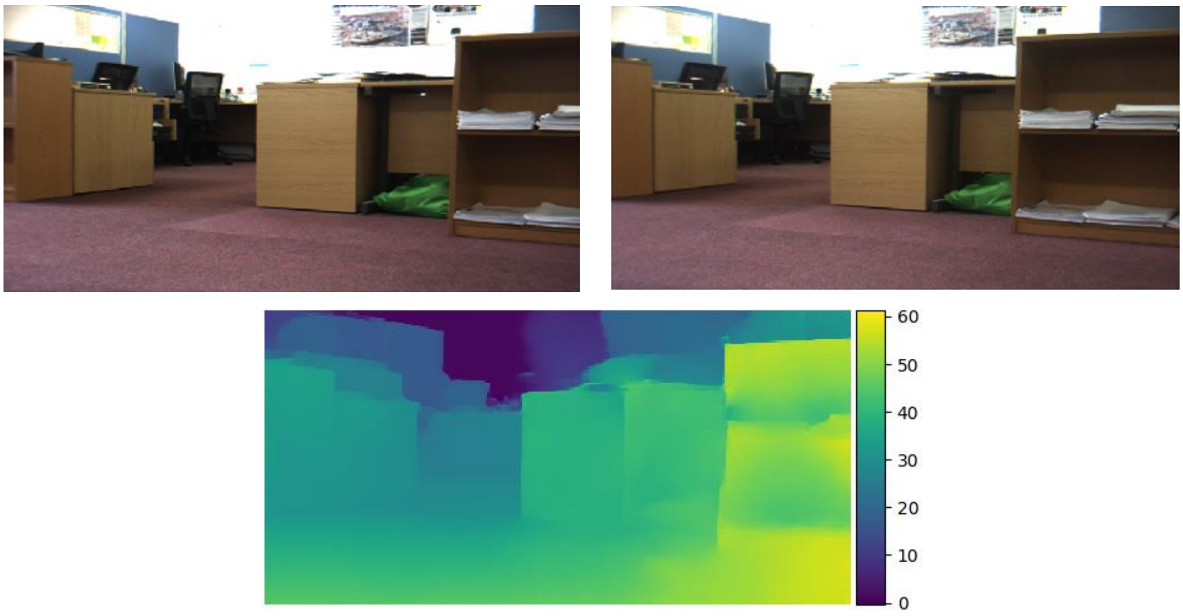



Figure 5.8: Disparity model qualitative test with different camera settings. Disparity Estimation for stereo image with stereo camera of 12cm baseline and 3.8mm focal length.

The second issue is the model suggestion for the zero disparity or the infinite distance. It can be seen that the model considered the wall as a zero disparity or faraway in the horizon, which reflects the convention of the dataset representations that the model trained on. This response has been encouraged by the wall surface smoothness. However, the model considered the posters that sticked on the same wall as objects and provided disparity readings for them, although they are at the same distance from the wall.

The reflection spots on a car or glass are problematic in motion estimation because they create distortions in the image level (high-intensity spike), affecting the image representation at the model input. As shown in Figure 5.9, the car image suffers light reflection, textureless, and transparent areas (car windows). All these challenging situations should be locally predicted based on the global semantics that the model should infer. The Figure *5.9* shows different problems that generally, the disparity estimation is experiencing, and the related results obtained.

In addition to the mentioned problematic issues, the thin structure cases like poles or smooth ground are also part of the global semantics that the model should understand. For example, in the first image of Figure 5.9, the thin structure is predicted. Furthermore, the white painting on the road could be exposed more if it is an object, but the model mapped it to the street level because it develops enough

knowledge about the shape of the road and its features during the training. In the second image, the sky usually appears on the top of the image with a specific color tone. Since this area represents points in the infinity, the model predicts very low disparity values.

Furthermore, the model can understand the transparent window as part of the car, although It shows objects with different depths through the car window. The model understands the universal shape of a car and completes its disparity map even in a highly ambiguous situation.

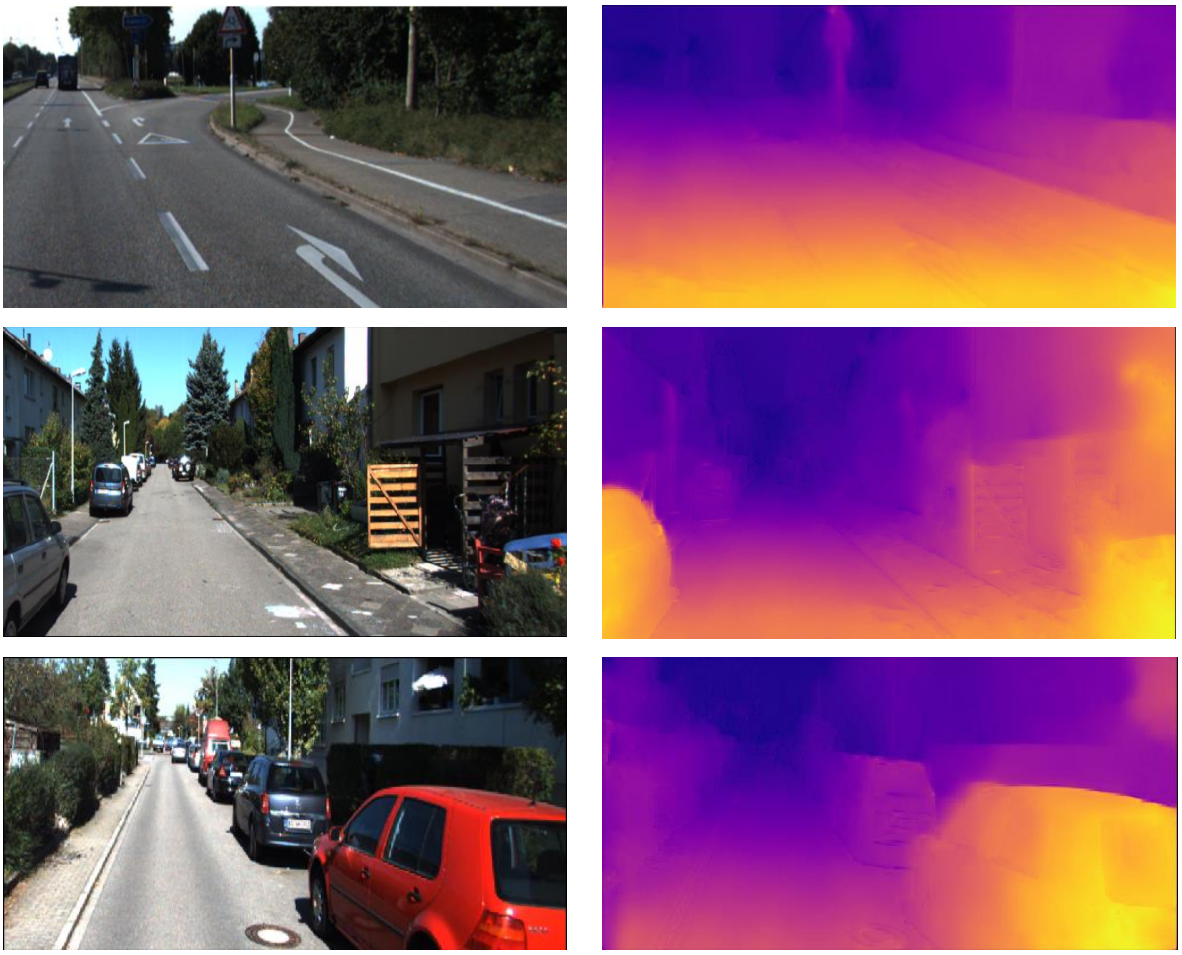


Figure 5.9: Disparity model qualitative test with Kiti dataset. Top: Image includes a thin structure and turning arrow on the ground. It can be seen that the model can identify the thin structure and able to understand that the drawings on the road are part of the road. Middle: The model can identify the pixel at infinity as the pixels of the sky areas. Bottom: global understanding of the object like cars such that the transparent and light reflected areas are ignored, and the car's complete structure is maintained.

It is worth analysing the dual-channel or occlusion aware model response regarding the error uncertainty at each pixel. An accumulation matrix is generated to obtain this analysis, which is technically an unnormalized 2D histogram estimation. The x-axis is the absolute error between the disparity prediction with respect to the ground truth. The y-axis is the standard deviation of the error at that particular pixel. The estimation is performed by generating the 100 error matrix from the test

images of the Kitti dataset. Each matrix represents the absolute error, and the standard deviation across these 100 reading per pixel.

Below the uncertainty estimation for two models. The first model is a single channel model trained without the training related to the occlusion error estimations (without Equations (5.9 and 5.12)). While the other model is the aware occlusion model trained with all the loss function reported (Equations 5.9-5.12) loss function used. Figure 5.10 shows the uncertainty estimation for the model with one channel. It can be seen that the distribution of the error property is horizontal, which is unfavorite characteristic (Mehltretter, 2020).

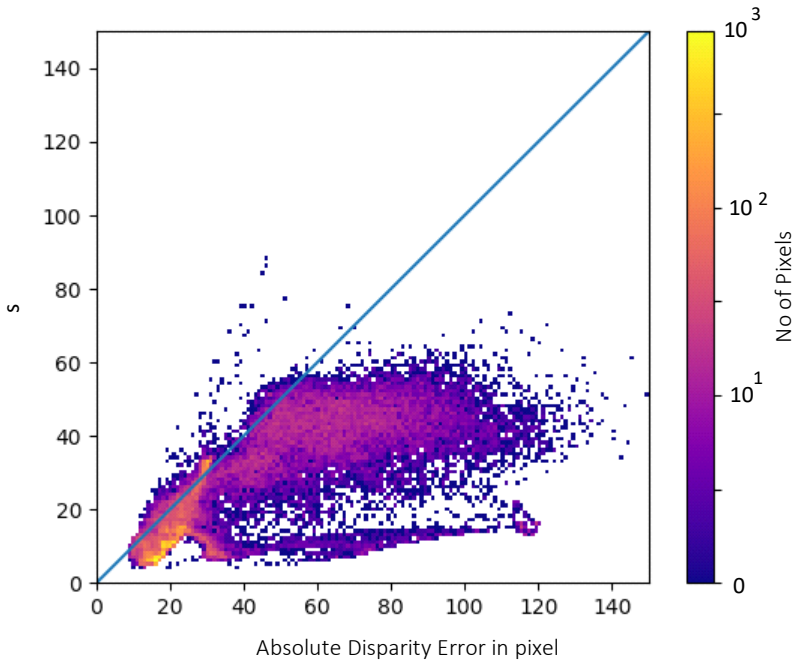



Figure 5.10: Single channel absolute error uncertainty test. Absolute error uncertainty relation on the Kitti dataset for a single channel model trained without occlusion aware loss function. The logarithmic colour scale encodes the number of pixels having the respective error and estimated standard deviation σ. It can be seen that the distribution of the pixel error is off the diagonal line. This distribution is an unwanted property *(Mehltretter, 2020)*.

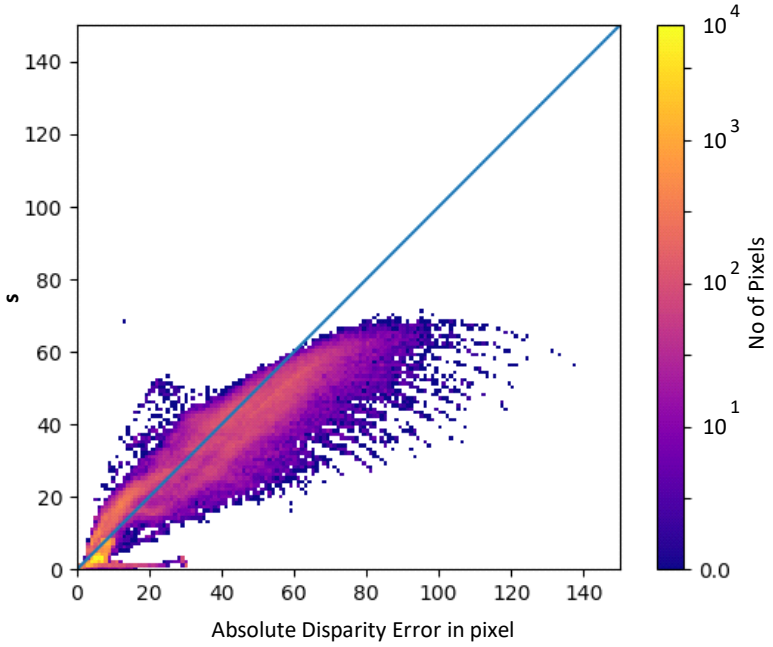



Figure 5.11: Dual channels absolute error uncertainty test. Absolute error uncertainty relation on the Kitti dataset for dual channel model. The logarithmic colour scale encodes the number of pixels having the respective error and estimated standard deviation σ. In case of optimal uncertainty estimation, all pixels should be aligned as close as possible to the diagonal line.

In Figure 5.11, the error property shows better characteristics as the accumulated absolute error is close to the diagonal line. These properties can reflect the related compensation to the occlusion ambiguity, which improves the model's uncertainty characteristics to the error.

## 5.8 Summary

A novel deep learning method for disparity estimation from a pair of rectified stereo images is presented. Two previously proposed network architectures have been combined to address the problem of unsupervised network training for the prediction of dense disparity maps. The method uses two prediction channels, with corresponding networks sharing weights, to estimate the forward/backward (left/right) disparities. Using the disparities consistency assumption, forward/backward occlusion masks are obtained, subsequently used to compute the data fidelity loss function more accurately. Two other components of the loss function have also been proposed. The final component of the loss function is introduced to regularize the disparity maps in the image uniform areas. There is little information to guide the estimation of the disparity from the image contents alone. The proposed model is trained end-to-end, and the results show improved prediction accuracy when tested on three popular datasets, FlyingThings3D, KITTI, and CityScapes. Although the network can be trained in a supervised manner, its main advantage is the ability to learn the regression model for disparity from data without the availability of the ground truth. It has been shown that the

model can achieve EPE of 2.5 pixels when trained without ground truth on the FlyingThings3D dataset, which is close to the value of 1.6 pixels, achieved in (Mayer, et al., 2016) with supervised learning. The model also shows the improved result for KITTI with 2.05 after fine-tuning.

## 5.9 Contribution

In this chapter, two novelty are presented. The first is related to the occlusion-aware network's proposed architecture that considers the occlusion in the training phase. The novelty is to combine two modified versions of the network reported in (Mayer, et al., 2016) in a new network. The network consists of two channels. One channel predicts the forward disparity while the other predicting the backword disparity. As a second contribution, a comparison between the results shows that the model achieved approximate linear uncertainty in the estimation of the disparity across the Kitti test set. This uncertainty property suggests that the model can inference disparity with an approximated compensation for the linear error function.

# Chapter 6 - Motion Field for Medical Image Registration

## 6.1 Introduction

Motion field estimation is an essential enabling step for many applications of computer vision. Chapter five demonstrated that 2D motion is a vital part of disparity computation leading to a scene depth estimation. In this chapter, the objective is to investigate novel approaches to 3D motion field estimation. 3D motion is critical for several biomedical applications of computer vision. This includes image registration and segmentation as well as detection and assessment of medical conditions. As this work is focused on developing novel computational approaches rather than solving specific clinical problems, for the description brevity of the biomedical applications, particulars of the 3D motion field estimation have been posing only in the context of volumetric data registration. The biomedical volumetric registration can be applied to many data modalities, including MRI, CT, or nuclear imaging (e.g., PET scans), enabling within and cross-modality inference. For example, patients or a subject may have multiple CT scans performed during and after treatment to assess a medical procedure's effectiveness.

Common deformable image registration algorithms require cost function optimization between two or more volumes (Sentker, et al., 2018). This requirement means that the algorithm needs to go through a computationally demanding and time-consuming optimization process for any new registration. These problems could cause a bottleneck for some procedures, e.g., radiation therapy (RT) workflows, including treatment planning. Clinical applications of the 4D CT data (3D spatial dimensions plus time) registration including semi-automated target volume and organ at risk contour propagation; assessment of motion effects on dose distributions (4D RT quality assurance, dose warping) (Rosu & Hugo, 2012) and 4D CT-based lung ventilation estimation and its incorporation into RT treatment planning (Yamamoto, et al., 2016). Image registration also plays a vital role in computer-aided diagnosis pipelines, radiation treatment planning, and image-guided interventions.

For the rest of this section, the CT lung registration was selected to demonstrate the proposed novel 3D motion field estimation's effectiveness. Lungs 3D CT volume registration is clinically important and a technically challenging problem. There are also many papers describing different lung registration

methods, which use common datasets for methods validation. Dataset availability makes the lung registration problem a suitable choice for the critical evaluation of a new registration method.

The key challenges that are typical for lung registration and that may not be found in other types of registration are:

- Large deformations that require an accurate and robust estimation of a large 3D motion/displacement.
- Rigid and deformable structures present in the same volume implicating that model should allocate each voxel to its structure container before applying for the necessary registration.
- Sliding motion of the lung against the chest wall during breathing needs to be identified by the model as these introduce a discontinuity in the motion field.

## 6.2 Literature Review

Many researchers are tackling medical image registration utilizing CNN. The authors in (Eppenhof & Pluim, 2018) presented a supervised approach using a sliding window to directly estimate the registration error using synthetically deformed CT images as well as a publicly available dataset containing landmarks annotations. Yang et al., 2017 also used the sliding window technique to estimate the 3D brain Double Inversion Recovery (DIR)'s deformation field. The main issue that could be mentioned about the patch-based learning is the lack of global information about the transformation that may not be adequately represented in small motion displacement but could appear in large displacement cases as in the CT scans case. In the context of Generative Adversarial Network (GAN), researchers perform multimodal registration using GAN (Hu, et al., 2018), Wasserstein GAN (WGAN) (Yan, et al., 2018), InfoGAN (Qin, et al., 2019), and Cycle GAN (Tanner, et al., 2018; Mahapatra, et al., 2018).

In (Hu, et al., 2018), vanilla GAN has been utilized to register Magnetic Resonance (MR) and 3D intra-procedural transrectal ultrasound (TRUS). They used supervised learning to train a discriminator part of the network by providing the generator's output and a simulated motion field. Authors (Dalca, et al., 2019) developed a probabilistic generative model and demonstrated their VoxelMorph network on a 3D brain registration task. Recently (Fang, et al., 2020), presented a deep learning model for unsupervised deformable thorax CT scan registration. The mentioned model has some similarities in the network design to the network proposed in this thesis, though with different training schemes.

In this work, the deformable registration problem is performed using 3D CNN. The required deformation is achieved using 3D motion field vectors to estimate the per-pixel displacement estimation in 3D using local and global representations by training with the full size of the CT volume of 256×256×128 voxel. A Bayesian version of the model is then introduced by converting it to probabilistic architecture (Gal & Ghahramani, 2016). This conversion is achieved by adding a discriminator to the network to train it in a GAN paradigm. The output of the model represents the 3D motion field vectors of 128×128×64×3 dimension. These vectors compensate for the deformation of the moving volume with respect to a reference volume. Unlike (Hu, et al., 2018), the discriminator is trained with the CT volumes during the network regularization. Introducing the LSGAN to the network improved the training and increased the network's performance on the test set.

## 6.3 Method

In this study, the deformable image registration is performed by designing the Transformation function to perform the mapping from the moving image to the reference image. Let $I_F(\boldsymbol{x}): \Omega_F \rightarrow \mathbb{R}$ and $I_M(\boldsymbol{x}): \Omega_M \rightarrow \mathbb{R}$ be two real value 3D images defined on their corresponding spatial domains $\Omega_F \subset \mathbb{R}^3$ and $\Omega_M \subset \mathbb{R}^3$ respectively. The task is to find the displacement field $\widehat{\boldsymbol{T}}: \Omega_M \rightarrow \Omega_F$, $\widehat{\boldsymbol{T}}(\boldsymbol{x}) = \boldsymbol{x} + \widehat{\boldsymbol{U}}(\boldsymbol{x})$, mapping pixels in the moving domain $\Omega_M$ to their corresponding pixels in the reference domain $\Omega_F$, where $\widehat{\boldsymbol{T}}$ represents the so-called displacement field. In the non-rigid registration framework, this motion field is usually estimated through solving the following optimization problem:

$$\widehat{\boldsymbol{T}} = \arg\min_{\widetilde{\boldsymbol{U}}}(sim(I_F, I_M(\widetilde{\boldsymbol{T}})) + \beta\, reg(\widetilde{\boldsymbol{T}})) \tag{6.1}$$

where $sim()$ is a dissimilarity function also called fidelity term depending on the observed data, and $reg()$ is called regularization term which reflects known or assumed properties of the displacement filed $\widehat{\boldsymbol{T}}$, typically encoding information about some form of $\widehat{\boldsymbol{T}}$ smoothness.

The quality of the registration, i.e., estimated function $\widehat{\boldsymbol{T}}$, can be assessed in several ways. In this work, the Target Registration Error (TRE) (Alansary, et al., 2019) is used. The TRE measure the displacement (here using the Euclidean distance) of $\widehat{\boldsymbol{T}}(\boldsymbol{x})$ from the true positions $\boldsymbol{T}(\boldsymbol{x})$ of the registered points (Eppenhof & Pluim, 2018):

$$TRE: \Omega_F \rightarrow \mathbb{R}^+: \boldsymbol{x} \rightarrow \left\| \boldsymbol{T}(\boldsymbol{x}) - \widehat{\boldsymbol{T}}(\boldsymbol{x}) \right\| \tag{6.2}$$

In practice, the true positions are not known, and a manually annotated set of target points (landmarks) is used as a surrogate of these positions. Therefore, manually annotated corresponding

landmarks $\boldsymbol{p_F} \in \Omega_F$ and $\boldsymbol{p_M} \in \Omega_M$ are used, and the TRE measures the Euclidean distance between $\boldsymbol{p_F}$ and the relocated, by the $\widehat{\boldsymbol{T}}$, $\boldsymbol{p_M}$ points.

### 6.3.1 Network

The convolutional network's implementation for the proposed deformable image registration consists of five 3D convolutional layers with stride 2 in the contractive part. The expander part consists of up-sampling followed by a corresponding 3D convolutional layer, see Table *6.1*. At each layer of the contractive part, the features are down-sampled across the three dimensions, x, y, and z. Similarly, at the expander part, the features are up-sampled at each layer across the three dimensions. The input to the network consists of the two volumes, the reference, and the moving volumes with a volume size of 128×256×256 as input size and the output size of 64×128×128 pixels, which is up-sampled later for the TRE computations. The size of the output has been chosen due to the limitation in the computational resources. For instance, the GPU used for this experiment was Nvidia TitanX with 12GB GPU memory. The activation function utilized in this implementation is leaky ReLU. At each layer of the expander, a reconstruction sampler is attached to map the moving volume to the reference volume using the transformation or the displacement function that the convolutional network is learning. This displacement consists of three maps corresponding to the x, y, and z components of the transformation function.

The discriminator consists of five blocks of VGG network layers (Simonyan & Zisserman, 2014) with an input of 64×128×128×2. The layer block consists of 3D convolution, down-sampling, batch normalization, and then leaky ReLU activation function. At the end of the discriminator, there are two Fully Connected Neural Network (FCNN) layers. In addition to the activation between these FCNN layers, dropout layers with a 50% dropout rate are attached. Then the discriminator is terminated with a linear regression layer to classify the input. The network's input can be either: the reference volume concatenated with the same volume after interpolating uniform noise in a range between [-0.01, 0.01] or a reference volume concatenated with the warped moving volume. In the context of the GAN model, the real class corresponds to the reference volume concatenated with its noisy modified version, while the fake class is represented by the reference volume concatenated with the moving volume. As the discriminator is starting to classify the fake input as a legitimate input, the generator performance improved. A simplified diagram representing the proposed architecture is shown in Figure 6.1.

| Contractive Part | | | | |
|---|---|---|---|---|
| Layer | Input | Output | No. of Features | Downscale |
| Conv1 | 256x256x128x2 | 128x128x64x32 | 32 | 2 |
| Conv2 | 128x128x64x32 | 64x64x32x64 | 64 | 2 |
| Conv3 | 64x64x32x64 | 32x32x16x128 | 128 | 2 |
| Conv4 | 32x32x16x128 | 16x16x8x256 | 256 | 2 |
| Conv5 | 16x16x8x256 | 8x8x4x512 | 512 | 2 |

(a)

| Expanding Part | | | | | |
|---|---|---|---|---|---|
| Layer | Input | Skip-Conn | Output | No. of Feature maps | Upscale |
| Up-Conv1 | 8x8x4x512 | 16x16x8x256 | 16x16x8x256 | 256 | 2 |
| Up-Conv2 | 16x16x8x256 | 32x32x16x128 | 32x32x16x128 | 128 | 2 |
| Up-Conv3 | 32x32x16x128 | 64x64x32x64 | 64x64x32x64 | 64 | 2 |
| Up-Conv4 | 64x64x32x64 | 128x128x64x32 | 128x128x64x32 | 32 | 2 |
| Up-Conv5 | 128x128x64x32 | | 128x128x64x3 | 3 | 1 |

(b)

| Discriminator | | | | |
|---|---|---|---|---|
| Layer | Input | Output | No. of Feature maps | Downscale |
| Conv1 | 128x128x64x2 | 64x64x32x8 | 8 | 2 |
| Conv2 | 64x64x32x8 | 32x32x16x16 | 16 | 2 |
| Conv3 | 32x32x16x16 | 16x16x8x32 | 32 | 2 |
| Conv4 | 16x16x8x32 | 8x8x4x64 | 64 | 2 |
| Conv5 | 8x8x4x64 | 4x4x2x128 | 128 | 2 |
| FCNN | 4x4x2x128 | 64 | 64 | |
| Drop-out | | | | |
| FCNN | 64 | 16 | 16 | |
| Dropout | | | | |
| FCNN | 16 | 1 | 1 | |

(c)

Table 6.1: Network layers details

### 6.3.2 Training Procedure

The training of the network consists of two phases; the Contractive-Expander phase and the GAN phase. The Contractive-Expander training phase includes the training of each convolutional layer of the Expander using 5000 iterations. It starts at the lowest resolution and gradually involving higher resolution levels of the expander, in each case trained using 5000 iterations. To estimate the error between the reconstructed or warped moving volume and the reference volume, the Generalized Charbonnier (GC) penalty function $g = (x^2 + \epsilon^2)^{\alpha}$ (Barron, 2019) is considered as the fidelity term, see Equation (6.3). This loss function is often used for optical flow and depth estimation. The function behaves as an L2 loss function close to zero to encourage smoothness, and L1 otherwise, to encourage robustness against outliers. It is also sometimes called the L1-L2 loss function.

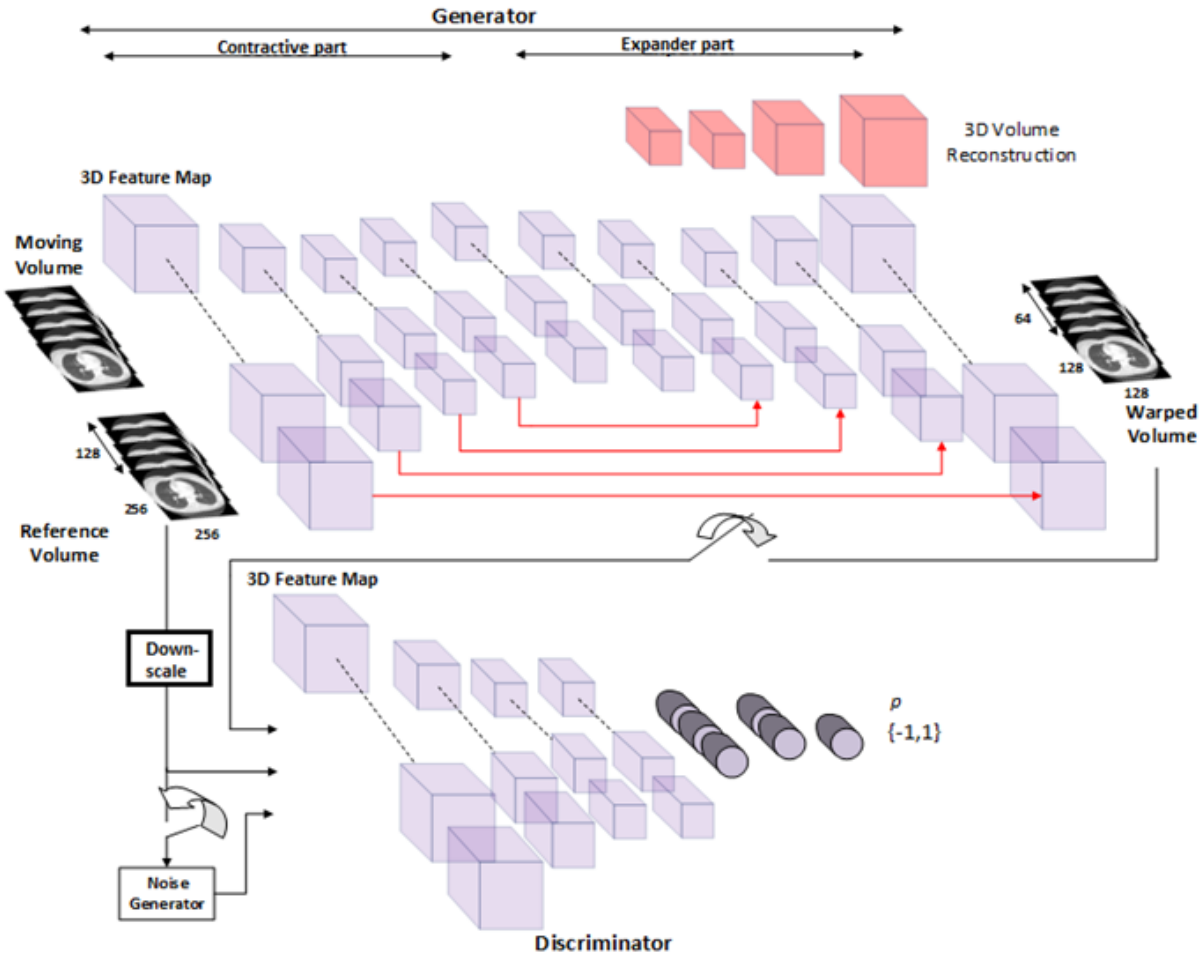



Figure 6.1: A simplified diagram representing the proposed network architecture. It consists of two parts: the contractive-expander as the generator and the discriminator.

The function is defined as:

$$sim = \ell_{GC} = \sum_{i,j,k} \left( \left( I_F(i,j,k) - I_{M\,warped}(i+u, j+v, k+l) \right)^2 + \epsilon^2 \right)^{\alpha} \tag{6.3}$$

where $\alpha$ and $\epsilon$ are chosen experimentally to be 0.7 and 0.0001 respectively, and $u$,$v$, and $l$ are elements of the displacement field $\widetilde{\boldsymbol{Q}}$. The adopted fidelity term implies the assumption that the corresponding true points in both volumes have the same intensity. However, this assumption is not exactly true in lung CT data because the intensity values of the corresponding points change due to changing air volume in the lungs in the inhale and exhale phases. A suitable intensity correction will be considered in a follow-up work, which can be learned independently or as a local Jacobians function. The current registration model hypothesizes that the adopted GAN implementation may implicitly compensate for that fidelity term modeling inaccuracy.

In addition to this loss function, while acting globally on the volume, the effect on regions with less image structure could create unambiguity in the displacement vector estimation (aperture problem). To address this ambiguity, a smoothing function that can minimize the multiple velocity scoring areas is utilized, Equation (6.4).

$$reg = \ell_S(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{l}) = \sum_{W,H,D} \rho(\nabla \widehat{\boldsymbol{Q}}) \tag{6.4}$$

where $\widehat{\boldsymbol{Q}}$ is the estimated deformable displacement, and the loss function is estimated across the volume $(W, H, D)$. $\rho$ is the realization of the GC function that is applied in these smooth sub-volumes like the organs with insufficient 3D image structures. This function will also enhance the smoothness of the moving voxels and close any gap within the representation of the displacement across the organ's boundaries. The total loss estimation is performed using the following weighting:

$$\ell_{Total} = \ell_{GC} + \beta \times \ell_S \tag{6.5}$$

The moving volume warpings performed using the method reported in (Jaderberg, et al., 2015) as in Figure 6.2. To reconstruct the volume, a bilinear interpolation is adopted.

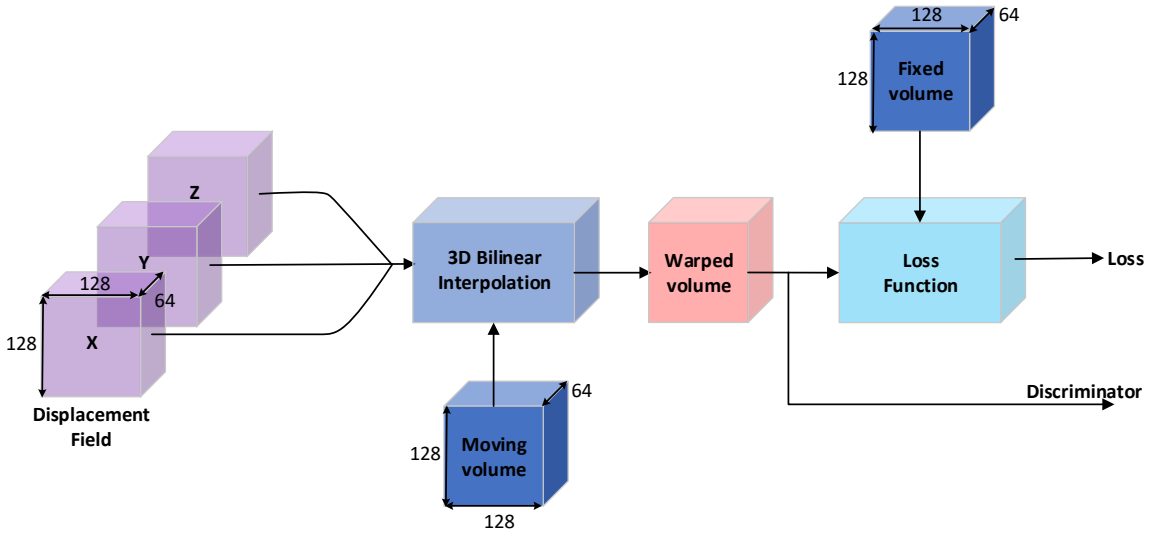



Figure 6.2: Diagram represents the estimation of the Loss function estimation

The next training phase (GAN training) starts while training the last layer of the GAN network generator. In this phase, the discriminator starts to be involved in the training. The dropout layers in the discriminator were quite effective in stabilizing the discriminator during the training of the model and mitigate against overfitting. The last layer is left without activation, and the resulting value is compared to {1} in the case of the real image and {-1} in the case of the fake image using LSGAN.

With the Least Square GAN, the discriminator is trained following the loss function of Equation (6.6) (Mao, et al., 2017):

$$\min_{D} V_{LSGAN}(Disc) = \frac{1}{2}\left(E_{x\sim P_{data}(x)}[(Disc(x)-1)^2] + E_{z\sim P_z(z)}[(Disc(G(z))+1)^2]\right) \quad (6.6)$$

here, $Disc(x)$ is the prediction of the discriminator when the input is the true reference volume, while $Disc(G(z))$ is the prediction of the discriminator when the input is the fake (which is the generator output or the warped moving volume $I_{M\ warped}$). Simultaneously the target of the generator is to learn the distribution $P_z(z)$ by sampling the input variable $z$ from the dataset distribution and map it through a differentiable network. The loss function to train the generator to perform the function mentioned above for this particular GAN is defined as in Equation (6.7):

$$\min_{G} V_{LSGAN}(G) = \frac{1}{2}E_{z\sim P_z(z)}[(Disc(G(z)))^2] \quad (6.7)$$

Using this arrangement, the model continued in its training using the optimization provided by the LSGAN network. It is worth mentioning that the discriminator network is stripped during the inferencing stage, and only the generator network is involved.

During the training, an Adam optimizer is adopted with a starting learning rate of 0.0001 that is reduced after every 5000 iterations by a factor of 1.4. Every 5000 iterations, the layer weights of the expander part are updated, starting from the lowest layer and going to the highest layer. The loss function is calculated using Equations (6.5) during this phase of training. When the training reaches the last layer of the expander part, the discriminator joins the training, and Equations (6.5) and (6.7) are utilized during the training of the generator in the following fashion:

$$L_{i>2000} = \ell_{Total} + \lambda \times \min_{G} V_{LSGAN}(G) \quad (6.8)$$

where $\lambda$=0.4.

In a typical implementation, the recommendation is to have a reasonably batch size larger than one sample. However, both the model and the volume sizes allowed only one sample per batch during the training. This meaning that the model is updating the weight after each sample (volume of 128×256×256×2). On the other hand, the sample of 128 images (in one volume) can still represent the statistical properties of the dataset to some extent.

### 6.3.3 Multi-resolution Image Registration

The training of the model is performed gradually in a multi-resolution loss estimation fashion. To achieve this loss estimation, the networks of Figure 6.2 has been provided at each resolution level. The resolutions start with 8x8x4 voxel dimension and duplicate at each stage and for five stages to reach 128x128x64 voxel dimension. In this registration task, where the expander layers are involved, the feature maps need to be similar in the voxels' volumic characteristics at a different resolution to provide consistent and stable features at the final stage. Therefore, this training scheme is vital as the network performance is affected equally by each stage of the network layers' performance. With this procedure, the network can be self-supervised without ground truth, and the backpropagation will be performed initially at the lower resolutions and train the network from that specific layer back to the first layer. As the training progress, after every 5000 iterations, the new layer with higher resolution will be involved in the training, and the backpropagation training will start from the newly introduced layer.

## 6.4 Dataset

For the training, both the CREATIS (Vandemeulebroucke, et al., 2012) and COPDgene (Castillo, et al., 2013) datasets have been utilised. The CREATIS dataset contains 4D CT scans for six subjects, each having ten volumes of 141 images that represent different stages of the inhale and exhale cycle. This volume size allows that when the sample is acquired from this dataset, a random start such that 128 images are available for data input. COPDgene consists of inhaling and exhaling volume sets of images for ten patients with around 100 images or slab per volume. At each acquisition, the sample has been resized to fit the network input using bilinear interpolation. Furthermore, during the data acquisition, the volumes were randomly swapped to be sometimes as fixed volume and other times as a moving volume, which increases the dataset's variability and improves the model regularization. Both datasets mentioned are used for training, and they were not considered during model testing or inferencing.

For the test set, the 4D DIR-LAB dataset (Castillo, et al., 2009) has been used. This dataset consists of ten subjects, with each entry includes ten volumes. This dataset's other important feature is that it includes 300 landmarks per volume for volume 1 and 5 annotated by experts. By conventions, the researchers map volume 5 to volume 1.

## 6.5 Results

In this work, the results were obtained for two cases. The first case is to train the model without GAN implementation. The training is performed using the reconstruction error obtained from the

difference between the reconstructed (warped) moving volume, and the reference volume, Equations (6.5). This case of training included 60K iteration. In the second case, the LSGAN network is implemented as in Figure 6.1, and the training included: a) Contractive-Expanding network trained for 20000 iterations using the warping loss, Equations (6.5). b) The discriminator joined the training for the rest of the 40K iterations. The error used to train the GAN during the 40K iterations is obtained from the classification of the combination of the reconstructed moving volume with the reference volume in one instance and the reference volume only in the second instance, Equations (6.6 & 6.7). Figure 6.3 shows landmarks for two volumes (Subject 8) of DIR-LAB before and after the registration. It is worth mentioning that the volume used for this experiment is quite large (128 slides) and covers a large space of the body. It is expected that the model provides similar quality of registration for the other parts of the body like the liver, stomach, and other organs under the pulmonary system and the diaphragm.

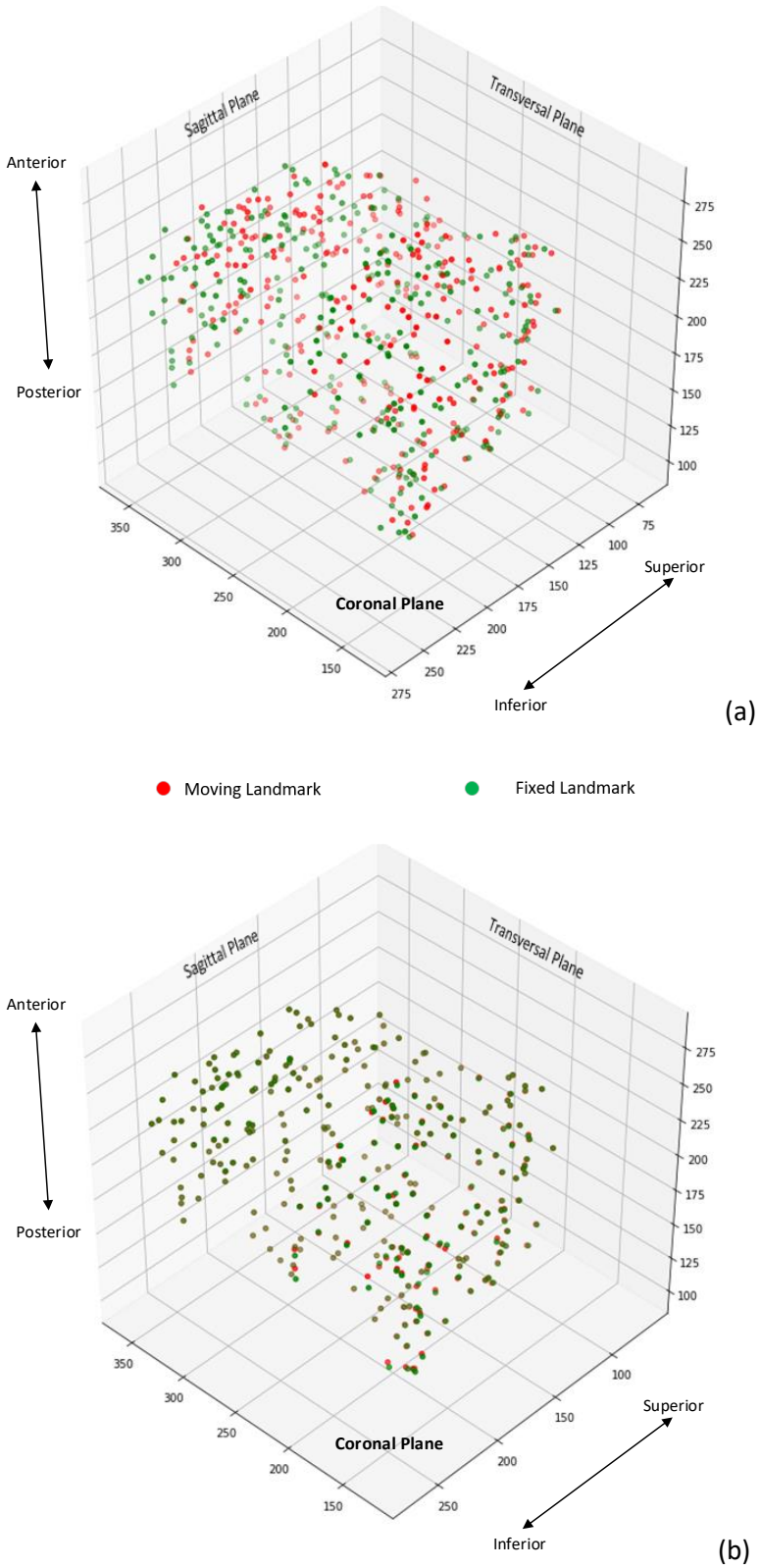



Figure 6.3:Annotated points of DIR-LAB, volume 8. (a): Before registration, (b): After registration, all axes in millimeters. The more color-saturated points represent points closer to the viewer.

Table 6.2 lists in the first column the landmark errors estimation before registration, followed by a registration obtained from (Papież, et al., 2014) implementing an iterative method, then, results obtained from training CNN model from (Fang, et al., 2020). Three professional observers performed the manual registration for annotation for 200 iterations (Castillo, et al., 2013) with their TRE is listed in the fifth column. The last two columns are for (The generator part only) network in one case trained without the LSGAN and with LSGAN. In the without LSGAN case, when the model consists of the generator network, regularization techniques of Equations (6.3-6.4) are utilized. The model achieved better performance than the expert annotation TRE. However, it could be noticed in the (TRE without the LSGAN) column that the maximum obtained error is still high for some volumes. Such behaviour appears when the model is not adequately regularized. Two reasons could be listed as the possible cause of this issue—first, the training set's limited size. Second, although the network uses L2 weight regularisation, and other smoothing loss functions, it lacks other regularization techniques listed in Section 3.4 like batch normalization, max-pool, etc.

To overcome the aforementioned regularisation issue, an LSGAN paradigm is involved in the network. With LSGAN, the average TRE showed an improvement of about 10% on the mean error. It could be seen that the obtained results with the usage of the LSGAN outperform all the previously reported results includes the expert observers TRE. For the LSGAN model (last column), the error was 0.44 average TRE, 0.43 SD, and 3.73 max error obtained for the landmark set of 300 points per volume. In addition to the improvement in the mean, both the SD and the maximum error improved by 50% compared to the model that was trained without the LSGAN. This improvement can be related to the fact that the model became less prone to overfitting. When the discriminator was incorporated into the network, the regularization effect due to the dropout and other regularisation techniques provided better network training conditions and improved the model's overall generalization.

| DIR-LAB | Before | After (Papież, et al., 2014) | After (Fang, et al., 2020) | Observers | Without LSGAN | LSGAN |
|---|---|---|---|---|---|---|
| **P1** | 3.89(2.77)(10.90) | 1.05(0.5) | 1.19(0.57) | 0.85 (1.24) | 0.28(0.12)(0.77) | 0.18(0.1)(0.67) |
| **P2** | 4.33(3.89)(17.69) | 1.08(0.6) | 1.08(0.60) | 0.70 (0.99) | 0.30(0.47)(5.22) | 0.35(0.34)(4.51) |
| **P3** | 6.94(4.04)(16.55) | 1.46(0.9) | 1.35(0.76) | 0.77 (1.01) | 0.17(0.10)(0.71) | 0.23(0.10)(0.84) |
| **P4** | 9.72(4.89)(20.25) | 2.05(1.5) | 1.63(0.99) | 1.13 (1.27) | 0.39(0.46)(4.53) | 0.41(0.36)(3.43) |
| **P5** | 7.34(5.52)(24.77) | 2.02(1.7) | 1.93(1.54) | 0.92 (1.16) | 0.47(1.11)(13.42) | 0.39(0.53)(5.08) |
| **P6** | 10.89(5.9)(27.59) | 2.48(1.8) | 1.94(1.49) | 0.97 (1.38) | 0.52(0.60)( 6.80) | 0.39(0.39)(2.10) |
| **P7** | 11.02(7.4)(30.63) | 2.78(2.3) | 1.98(1.46) | 0.81 (1.32) | 0.54(0.83)( 11.73) | 0.52(0.76)(8.92) |
| **P8** | 14.99(8.9)(30.57) | 3.96(3.8) | 3.97(3.91) | 1.03 (2.19) | 0.97(1.69)( 16.16) | 0.97(0.69)(4.61) |
| **P9** | 7.91(3.97)(15.76) | 1.89(1.2) | 1.92(1.44) | 0.75 (1.09) | 0.37(0.51)( 4.41) | 0.22(0.21)(1.67) |
| **P10** | 7.30(6.3)( 27.79) | 2.35(2.5) | 1.95(2.40) | 0.86 (1.45) | 0.88(1.37)(13.68) | 0.77(0.78)(5.45) |
| **Mean** | 8.43(5.48)(22.19) | 2.11(0.9) | 1.89 (1.94) | 0.879(1.31) | 0.49(0.72)(7.84) | 0.44(0.43)(3.73) |

Table 6.2:Target Registration Error (TRE) for DIR-LAB dataset. Some column consists of three values, Error Mean, (Error SD), and (Maximum Error), while others reported only the mean and the SD. The second column (Before Registration) is the error of the landmarks between the reference and the moving volumes. The third and fourth columns are after registration of reference *(Papież, et al., 2014)* (iterative method) and of reference *(Fang, et al., 2020)* using CNN. Column 5 is referred to the manual registration performed by professional observers reported in *(Castillo, et al., 2013)*. Columns 6-7 are the network of Figure (6.1) trained without GAN and with LSGAN, respectively. About 15% improvement in the registration obtained after training with the GAN paradigm.

Furthermore, it has been noticed that part of the performance improvement when the LSGAN is involved is related to the lower maximum error obtained. It can be seen in Figure 6.3. that the max error appears when a large displacement is required. Having a lower maximum error in the estimation of the LSGAN tells that the model's responses to large displacement improved, which is usually challenging in the deformable registration problem.

The model can perform the dense 3D motion field estimation in about 19.3msec using INVIDIA TITAN X Pascal graphical interface with 16GB computer memory and Intel i7 CPU.

Figure 6.4 is provided to show the TRE histogram for each volume case of the DIR-LIB dataset. It can be seen that a small SD of the histogram reflects a good generalization of the model.

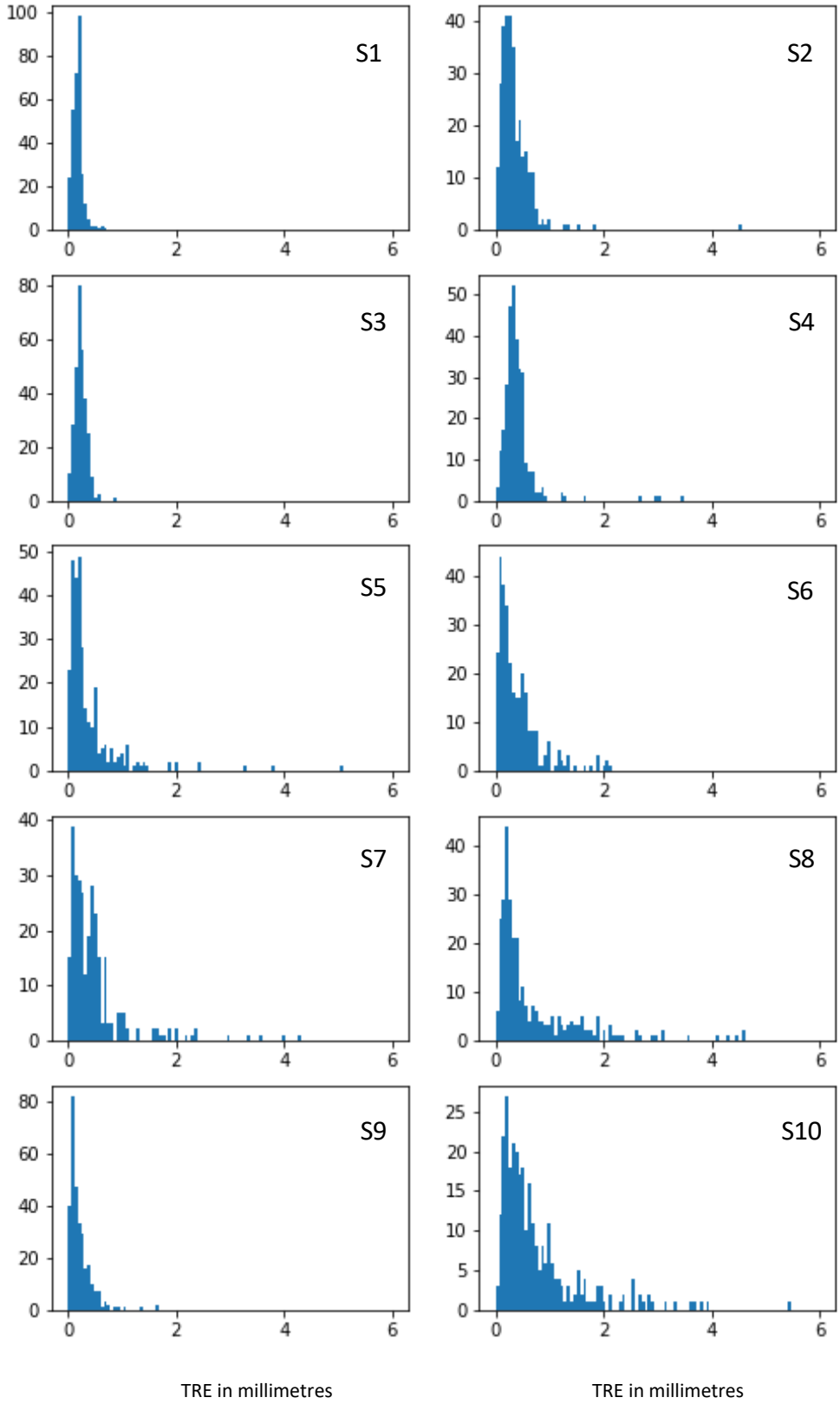



Figure 6.4: Histogram of the TRE for the ten subject's volumes of DIR-LAB dataset. The horizontal axis shows the Error values.

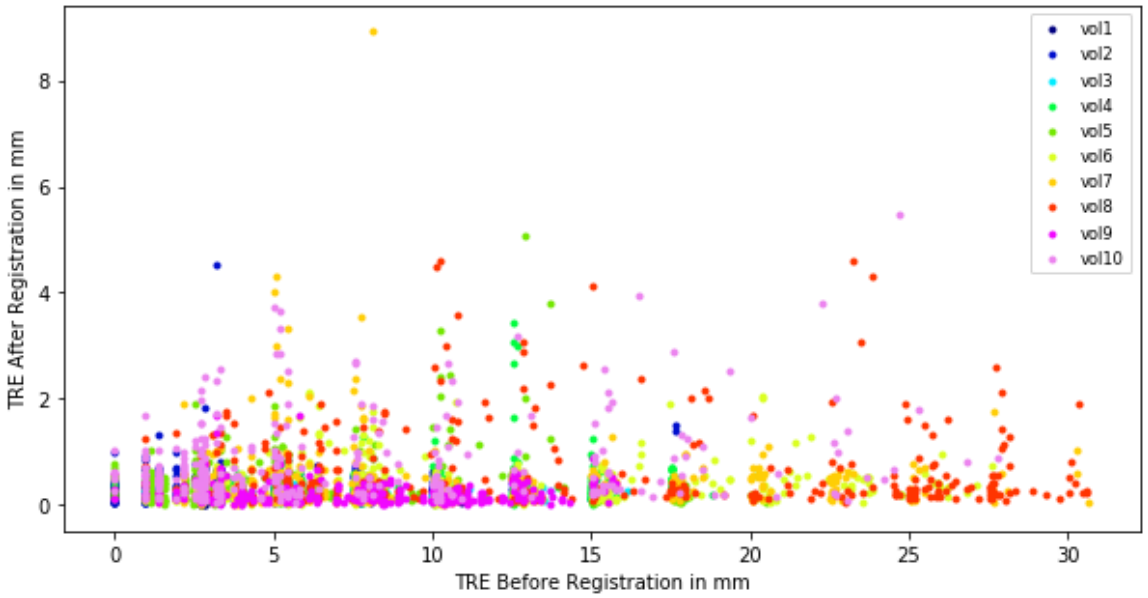
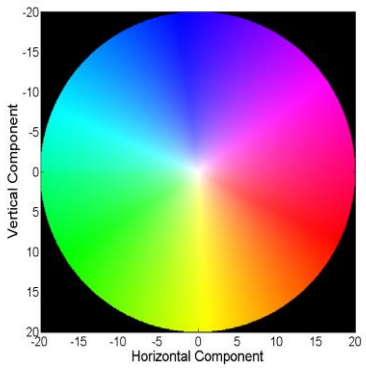



Figure 6.5: Distribution of the TRE before and after Registration

Figure 6.5 shows the TRE distribution of the landmark for the model with the LSGAN for the ten volumes before and after the registration. It can be seen that volumes 6-8 are experiencing high TRE before the registration; however, the model adequately reduced it.

For the reasons above, further investigation about case 8 of the DIR-LAB dataset has been provided in Figure*6.6*. It shows a sample of the cross-sections of the 3D displacement field of the model for that case. The network accommodates large displacements to perform the transformation (reflected as a saturated colour). It also shows that the model can implicitly segment the volume into different anatomical organs and impose relevant constraints on the resulting displacement field. Although it is not clear how the model can develop this knowledge, the model could potentially group voxels into different classes based on the intensity and belonging to the bone as a rigid organ or other non-rigid organs and transform each voxel set accordingly. Voxels related to rigid objects (bones) should be transformed based on rigid transformation. When the voxel belongs to a non-rigid class, an elastic transformation could be applied. This concept can be seen on the boundaries around the bones. The network also recognizes the non-rigid organs and transforms their voxel differently with different displacement values, although they are very close to each other.



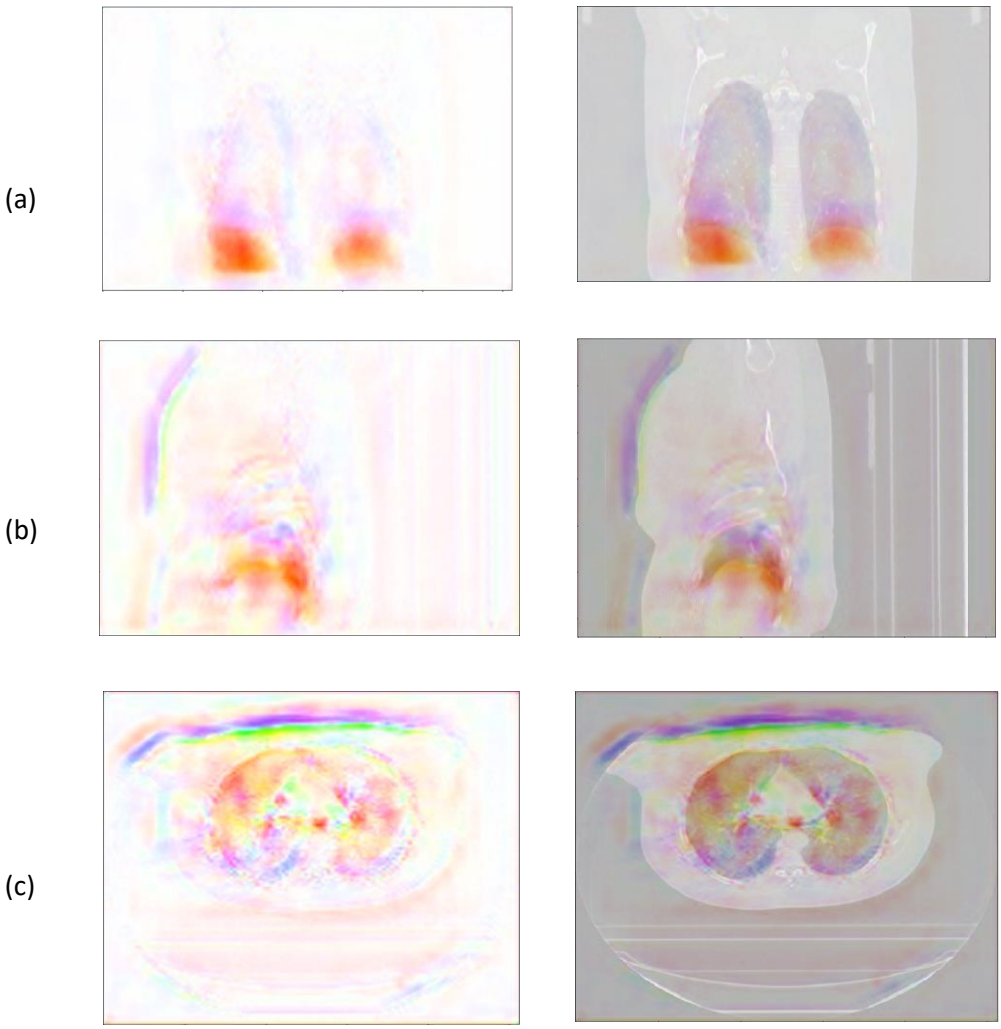




Figure6.6:Optical flow of the 3D CT scan cross sections. (a) Coronal plan. (b) Sagittal plan. (c) Transversal plan. Note, only $u, v$ representations show in the optical flow.

## 6.6 Summary

This chapter describes a novel deformable volume registration method using contraction-expansion CNN, configured without and with a discriminator sub-network within a GAN training framework. The proposed architecture is evaluated on the DIR-LAB dataset with registration performed on exhale – inhale sequences of lung CT scans. The results show that the proposed method achieves better performance when trained with the discriminator sub-network in the GAN training regime and excel the human performance. The use of the discriminator in the GAN like-training improves the performance of the network by approximately 10%, with the state-of-the-art TRE mean error of 0.44mm and SD of 0.43mm. These results are state of the art compared to the previously reported methods evaluated on the same dataset. The inferencing time became an essential factor in implementing a successful online inferencing system. Although the inference time has not always been clearly reported in previously published work, it is worth mentioning that the dense motion field's estimation when using the proposed method enables an estimate of the entire 3D registration motion field within 19.3ms.

## 6.7 Contribution

The contribution related to this work appears in three areas. First, instead of training the model with small volume patches or using ground truth landmarks, a DL model trained end-to-end on the full size of the volume in an unsupervised fashion. The entire volume access allows a better understanding of the global data representation and motion field resolutions (64x128x128) voxels. The second contribution can be referred to the model dimensionally, architecture, and depth used for this problem, which led to outstanding results as state of the art in this field. The final contribution is referred to as using the GAN as an option in the training process to improve the performance as a regularisation technique, which can be used in other previously reported works in different fields in the DL field.

# Chapter 7 - CONCLUSION AND FUTURE WORKS

## 7.1 Conclusion

This thesis explores motion detection and estimation, which are essential problems in computer vision. One of the key observations which can be drawn from the reported work is that the end-to-end training of deep learning models has emerged as a successful technique for computer vision problems. In case of the eyeblink detection problem, the network trained in the end-to-end fashion outperforms processing pipeline consisting of two networks, spatial and temporal, trained independently. For the other problems considered in this thesis, models which use the end-to-end learning outperform other approaches including hand-engineered and machine learning models developed previously. This learning paradigm also reduces the designer effort and generally perform better at optimising cost function.

### 7.1.1 Eye Blink Detection

The dataset used for the eyeblink detection has been deliberately arranged in such a way that there is no clear correspondence in the applied training video sequences. Even though the images in the training set depict different subjects and the eye location could be significantly different in each image, the model was able to establish correspondence at the deeper layers of a CNN and before the data passed to LSTM have well established spatiotemporal correspondence.

Motion detection in the video can be seen as a 3D problem, with two spatial dimensions and one temporal dimension. Feature learning in in this case can be constructed as a 3D convolution, if not simplified. Solving this problem by separating the spatial and temporal dimensions and then aggregating the corresponding features could produce similar effective results at much lower computational cost.

Evidence has shown that although image frames in the model shown in Figure *4.9*(c) are processed first using spatial convolution, the spatial characteristics of the generated features possess motion effects like blur. This can be explained as the effect of the other frames in the model when they are passing simultaneously across parallel channels of a model with shared weights. Here, the backpropagation will encode the motion information in the model features before the LSTM layers.

Since the weights of the 2D convolutional layers are shared, each channel will develop similar blurry activation but different in time space on the input frame. This can provide a consistent input to the LSTM layers and improve the training.

### 7.1.2 Disparity Estimation

Stereovision disparity estimation is a challenging task due to the large number of estimated parameters specially in the autonomous vehicle navigation datasets like Kitti. In this work, assuming rectified images are available, the disparity can be considered as a one-dimension motion estimation in the horizontal direction, with the corresponding dense displacement filed estimated using a DL model. Among many issues, occlusion that it common stereovision problem introduces error in the loss function computations, leading to substandard training particularly for the unsupervised learning.

The qualitative results showed an acceptable result, showing clear representation of the disparity even for the objects deep within the scene. This type of behavior shows consistent response of the network and good level of the image interpretation by the network. In the same time, the quantitative results showed better performance that previously reported methods. Furthermore, analysis showed that the model trained with the occlusion aware setting provides more error consistency, which could be compensated for, if needed.

The occlusion area appears bigger when the objects are closer to the camera and these areas contribute more in diverting the learning algorithms from the optimum solutions. Usually, this ambiguous error areas introduces stronger effect on the real error compared to the other areas where the disparity values are lower. Reducing this ambiguous error due to these big disparity values in the network improves on average the estimation of the disparity, across the image, including the disparities of the objects deeper in the scene.

### 7.1.3 3D CT scan image registration

In 3D CT scan image registration, the key challenges for lung registration include (i) large deformation, (ii) existence of rigid and deformable structures, (iii) sliding motion between different organs within the thorax volume. Each of these characteristics requires a specific transformation property, and the solution should accommodate all these requirements.

Another important choice during the network implementation was the loss functions selected to train the model in addition to the Generalized Charbonnier loss function utilized for both the fidelity term and the smoothness term of the loss function. This combination shows important results on the

network performance even before using the LSGAN training. Unlike other studies that used loss functions that are different in their characteristics when they employed the GAN paradigm in their implementation, choosing the LSGAN loss function is a close attribute to the loss function used to train the Contractive-Expander network. This choice allows the generator network to receive an update in the weights within the same properties as the update received from the GC loss function.

## 7.2 Contribution

To summarise, the contribution of this project includes:
Two novel DL models have been developed as solutions for the motion detection application found in the eyeblink problem. Each model addresses a specific formulation of this motion detection problem. The first model used separate CNN to extract latent features from images and RNN to extract the temporal features and joined as a one CNN-RNN pipeline solution trained in two stages. In the second novel solution, a TDCNN-LSTM model has been trained end-to-end for eyeblink detection. The second solution showed state-of-the-art to performance when compared to all previously reported solutions.

**Disparity Estimation**

In the 2D motion estimation, a novel model that consists of two channels trained simultaneously to estimate the forward and backward disparity maps has been proposed. This arrangement allowed to develop a model that can be used to estimate the occlusion masks. The developed occlusion masks have been utilised to develop an occlusion aware loss function during the training. Due to the shared training tasks of the forward and the backward disparity estimation, the model showed better regularisation and the capacity to generalise better for unseen data. The developed model showed linear error characteristics during the quantitative test when compared with other models trained with occlusion agnostic loss function.

**3D Image Registration**

A novel deep learning model for registration of 3D images has been designed and implemented. The model addresses the challenge of a large volume of data that needs to be processed. The model has been trained to develop a smooth motion field that can be upscaled with the appropriate interpolation functions to match the size of the original data volume. The model showed superior performance when evaluated on a benchmark dataset. Moreover, the model has been further extended by including embedded GAN, and showed further improvement in the estimation of the 3D motion field. The novel training regime, with embedded GAN, improves the results suggesting the GAN's capacity

to improve the regularisation of the model. Both versions of the proposed model showed the state-of-the-art performance for 3D deformable registration problem, improving on the results achieved using manual registration controlled by a trained expert.

## 7.3 Future Work

The work carried on in this thesis has many aspects of future work that can be important to be pointed to. Like most other research work, this thesis raises more questions than answers and opens for other research ideas. In addition to the fact that the algorithms presented here could be improved, they could represent essential ideas for further work in other applications and future researches that touch exciting aspects.

### 7.3.1 Structure from Motion

Structure from Motion (SfM) is an old problem in computer vision that allows to reconstruct the scene from a video sequence. The images of the video usually contain correspondence. However, they may have different camera pose with the evidence that images with similar structure or characteristics similar to the eye blink detection dataset experiment, a model can be trained to reconstruct an object in 3D based on a group of video images. The model is required to be trained in a supervised paradigm, and the 2D to 3D reconstruction needs to be included in the model.

### 7.3.2 Augmented Reality

The augmented reality is a way to introduce a digitally processed object or scene to the real-world scene that we are experiences by our eye. In other words, the augmented reality adds unreal objects to our perceived sight. Developing wearable glasses that can overlay augmented data over the current scene like advertisements as a person approaches a place or while playing, could be very near-future technology. The key requirement of this technology is the depth data that can allow an accurate overlay of the digital contains on the real scene such that the virtual world will not show conflict in the contents with the real world.

### 7.3.3 Regularization using Discriminator

In this work, the GAN paradigm showed improvement in the system performance with a respectable value of 10% in the registration problem. The future work will be focused on a better understanding of the role of the discriminator and whether it could be used as a regularization technique that could

compensate for other techniques that hard optimize as they are part of the total network training process compared to the training of the discriminatory which is trained separately.

Furthermore, has been mentioned in Chapter 6, further investigation will be conducted to consider the effect of correcting the volume intensity by learning explicit features they can be learned independently or as a local Jacobians function.

# Appendix A

## 1. Precision and Recall

Precision is the measure of exactness or the percentage of samples labeled as positive and actually such, whereas recall is the measure of completeness or the percentage of positive samples that labeled as such.

$$Precision = \frac{True\ Positive}{True\ Positive+False\ Positive} \quad , \qquad Recall = \frac{True\ Positive}{True\ Positive+False\ Negative}$$

A precision score 1.0 is a perfect score for a class which is by definition means that the model succeeded to classify each and every sample that belong to the class. However, it is not telling how many of the samples that doesn't belong to that class that mislabeled as sample from that same class.

On the other hand, a recall of score 1.0 means a perfect score that implies that every sample of a class was labeled as suggested, but it does not give further information about eh incorrectly classified samples that belonging to the that class.

## 2. F1 Score

F1 score, and other F scores, comes to solve these ambiguities in the previously mentioned scores, precision and recall. F scores is defined as:

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

The F1 score is the harmonic mean of precision and recall. In the previous equation, the F score gives both of the Precision and Recall the same weight.

## 3. Target Registration Error (TRE)

The error of the target fiducial markers following registration that can be expressed as:

$$TRE = \|T(\boldsymbol{p}_m) - \boldsymbol{p}_r\|$$

where $T(\boldsymbol{p}_m)$ is the mapping of the landmark pixel after the registration. The landmark in this context is a vector of the 3D space coordinates $(x, y, z)$. To calculate the total error for $n$ landmarks the estimation is performed as below:

$$TRE = \sqrt{\sum_{i=1}^{n} (T(\boldsymbol{p}_{m_i}) - \boldsymbol{p}_{r_i})^2}$$

# Appendix B

## Co-authored Papers

1. Anas, E.R., Henriquez, P. and Matuszewski, B.J., 2017. Online Eye Status Detection in the Wild with Convolutional Neural Networks. In *VISIGRAPP (6: VISAPP)* (pp. 88-95).
2. Anas, E.R. and Matuszewski, B.J., 2018. Dynamic State Recognition Using CNN-RNN Processing Pipeline. *International Journal of COMADEM*, *21*(3).
3. Anas, E.R., Guo, L., Onsy, A., and Matuszewski, B.J., 2019, June. Scene disparity estimation with convolutional neural networks. In *Multimodal Sensing: Technologies and Applications* (Vol. 11059, p. 110590T). International Society for Optics and Photonics.
4. Anas, E.R., Onsy, A., and Matuszewski, B.J., 2020. CT Scan Registration with 3D Dense Motion Field Estimation Using LSGAN. InAnnual Conference on Medical Image Understanding and Analysis. MIUA2020.

# Appendix C

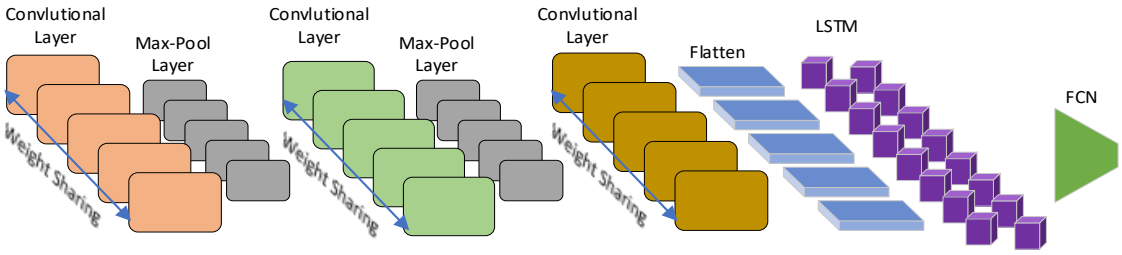



Figure (App.C) shows a Time-Distributed Convolutional Neural Network with an LSTM example.

# References


Aach, T. & Kaup, A., 1994. Disparity-based segmentation of stereoscopic foreground/background image sequences. *IEEE Transactions on Communications,* Volume 42234, pp. 673-679.

Aitken, A. et al., 2017. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. p. arXiv preprint arXiv:1707.02937.

Alansary, A. et al., 2019. Evaluating reinforcement learning agents for anatomical landmark detection. *Medical image analysis,* Volume 53, pp. 156-164.

Aloysius, N., Geetha, M. N. & Geetha, M., 2017. A review on deep convolutional neural networks. *In 2017 International Conference on Communication and Signal Processing (ICCSP)*, April, pp. 0588-0592.

Alzughaibi, A., Hakami, H. & Chaczko, Z., 2015. Review of human motion detection based on background subtraction techniques. *International Journal of Computer Applications,* 122(13).

Anas, E., Henriquez, P. & Matuszewski, B., 2017. Online Eye Status Detection in the Wild with Convolutional Neural Networks. *In VISIGRAPP,* 6(VISAPP), pp. 88-95.

Anas, E. & Matuszewski, B., 2018. Dynamic State Recognition Using CNN-RNN Processing Pipeline. *International Journal of COMADEM,* 21(3).

Arora, S., Bhaskara, A., Ge, R. & Ma, T., 2014. Provable bounds for learning some deep representations. *In International Conference on Machine Learning,* January.pp. 584-592.

Baker, S. & Matthews, I., 2004. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision,* 56(3), pp. 221-255.

Baltrušaitis, T., Robinson, P. & Morency, L., 2016. Openface: an open source facial behavior analysis toolkit. *In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV),* pp. 1-10.

Barron, J., 2019. A general and adaptive robust loss function. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4331-4339.

Bengio, Y., 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning,* 2(1), pp. 1-127.

Bengio, Y. C. A. a. V. P., 2012. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR,* abs/1206.5538(1).

Bengio, Y., Courville, A. & Vincent, P., 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence,* 35(8), pp. 1798-1828.

Bengio, Y., Léonard, N. & Courville, A., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint,* p. arXiv:1308.3432.

Berg, A., Deng, J. & Fei-Fei, L., 2011. *Large scale visual recognition challenge 2010.* [Online] Available at: http://www. image-net. org/challenges/LSVRC/2010/index

Bergasa, L. et al., 2006. Real-time system for monitoring driver vigilance. *IEEE Transactions on Intelligent Transportation Systems,* 7(1), pp. 63-77.

Bock, S., Goppold, J. & Weiß, M., 2018. An improvement of the convergence proof of the ADAM-Optimizer. Issue arXiv preprint arXiv:1804.10587.

Borza, D., Itu, R. & Danescu, R., 2018. In the Eye of the Deceiver: Analyzing Eye Movements as a Cue to Deception. *Journal of Imaging,* 4(10), p. 120.

Cao, Y., Wu, Z. & Shen, C., 2017. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology,* 28(11), pp. 3174-3182.

Castillo, R. et al., 2013. A reference dataset for deformable image registration spatial accuracy evaluation using the COPDgene study archive. *Physics in Medicine & Biology,* 58(9), p. 2861.

Castillo, R. et al., 2009. A framework for evaluation of deformable image registration spatial accuracy using large landmark point sets. *Physics in Medicine & Biology,* 54(7), p. 1849.

Chatterjee, I. & Sharma, A., 2018. Driving Fitness Detection: A Holistic Approach For Prevention of Drowsy and Drunk Driving using Computer Vision Techniques. *n 2018 South-Eastern European Design Automation, Computer Engineering, Computer Networks and Society Media Conference (SEEDA_CECNSM),* pp. 1-6.

Chen, Z., Wu, C. & Tsui, H., 2003. A new image rectification algorithm. *Pattern Recognition Letters,* 24(1-3), pp. 251-260.

Clevert, D., Unterthiner, T. & Hochreiter, S., 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv,* p. preprint arXiv:1511.07289.

Cordts, M. et al., 2016. The cityscapes dataset for semantic urban scene understanding. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 3213-3223.

Dahlhaus, R., 1996. On the Kullback-Leibler information divergence of locally stationary processes. *Stochastic processes and their applications,* 62(1), pp. 139-168.

Dalca, A., Balakrishnan, G., Guttag, J. & Sabuncu, M., 2019. Unsupervised learning of probabilistic diffeomorphic registration for images and surfaces. *Medical image analysis,* Volume 57, pp. 226-236.

Dechter, R., 1986. Learning while searching in constraint-satisfaction problems. *aaai.org.*

Dinh, H., Jovanov, E. & Adhami, R., 2012. Eye blink detection using intensity vertical projection. *In International Multi-Conference on Engineering and Technological Innovation,* Volume IMETI 1.

Divjak, M. & Bischof, H., 2009. Eye Blink Based Fatigue Detection for Prevention of Computer Vision Syndrome. *In MVA ,* pp. 350-353.

Dosovitskiy, A. et al., 2015. Flownet: Learning optical flow with convolutional networks. *In Proceedings of the IEEE international conference on computer vision,* pp. 2758-2766.

Drutarovsky, T. & Fogelton, A., 2014. Eye blink detection using variance of motion vectors. *In European Conference on Computer Vision,* Springer(Cham), pp. 436-448.

Du, Y., Ma, P., Su, X. & Zhang, Y., 2008. Driver fatigue detection based on eye state analysis. *In 11th Joint International Conference on Information Sciences. Atlantis Press.*

Eigen, D., Puhrsch, C. & Fergus, R., 2014. Depth map prediction from a single image using a multi-scale deep network. *In Advances in neural information processing systems,* pp. 2366-2374.

Eppenhof, K. & Pluim, J., 2018. Error estimation of deformable image registration of pulmonary CT scans using convolutional neural networks. *Journal of medical imaging,* 5(2), p. 024003.

Fang, Q. et al., 2020. A FCN-based Unsupervised Learning Model for Deformable Chest CT Image Registration. *In 2019 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC),* Volume IEEE, pp. 1-4.

Fan, Z. et al., 2018. Adaptive crowd segmentation based on coherent motion detection. *Journal of Signal Processing Systems,* 90(12), pp. 1651-1666.

Fazli, S. & Esfehani, P., 2012. Automatic Fatigue Detection Based On Eye States. *In Proc. of Int. Conf. on Advances in Computer Engineering.*

Fernando, B. & Gould, S., 2016. Learning End-to-end Video Classification with Rank-Pooling. *ICML.*

Fischer, A. & Igel, C., 2014. Training restricted Boltzmann machines: An introduction. *Pattern Recognition,* 47(1), pp. 25-39.

Fitzgibbon, A. & Fisher, R., 1996. A buyer's guide to conic fitting. *University of Edinburgh, Department of Artificial Intelligence,* pp. 513-522.

Fogelton, A. & Benesova, W., 2016. Eye blink detection based on motion vectors analysis. *Computer Vision and Image Understanding,* Volume 148, pp. 23-33.

Fogelton, A. & Benesova, W., 2018. Eye blink completeness detection. *Computer Vision and Image Understanding,* Volume 176, pp. 78-85.

Foley, J. et al., 1994. *Introduction to computer graphics.* s.l.:(Vol. 55). Reading: Addison-Wesley.

Gal, Y. & Ghahramani, Z., 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *In international conference on machine learning,* June.pp. 1050-1059.

Gao, L. et al., 2017. Video captioning with attention-based LSTM and semantic consistency. *IEEE Transactions on Multimedia,* 19(9), pp. 2045-2055.

Geiger, A., Lenz, P., Stiller, C. & Urtasun, R., 2013. Vision meets robotics:The KITTI dataset.. *The International Journal of Robotics Research,* 32(11), pp. 1231-1237.

Gibson, J. D. & Bovik, A., 2000. *Handbook of Image and Video Processing.* Jerry D. Gibson;Al Bovik ed. Orlando: Academic Press, Inc.

Girshick, R., 2015. Fast r-cnn. *Proceedings of the IEEE international conference on computer vision,* pp. 1440-1448.

Godard, C., Mac Aodha, O. & Brostow, G., 2017. Unsupervised monocular depth estimation with left-right consistency. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 270-279.

Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep learning.* s.l.:MIT press.

Grauman, K. et al., 2003. Communication via eye blinks and eyebrow raises: Video-based human-computer interfaces. *Universal Access in the Information Society,* 2(4), pp. 359-373.

Graves, A., Mohamed, A. & Hinton, G., 2013. Speech recognition with deep recurrent neural networks. *IEEE international conference on acoustics, speech and signal processing,* pp. 6645-6649.

Group, F. R. W., n.d. *Talking Face Video.* [Online]
Available at: http://www-prima.inrialpes.fr/FGnet/data/01-TalingFace/talking_face.html
[Accessed 28 November 2020].

Hao, J., Li, C., Kim, Z. & Xiong, Z., 2012. Spatio-temporal traffic scene modeling for object motion detection. *IEEE Transactions on Intelligent Transportation Systems,* 14(1), pp. 295-302.

Hartley, R. & Zisserman, A., 2003. *Multiple view geometry in computer vision.* s.l.:Cambridge university press.

He, K. & Sun, J., 2015. Convolutional neural networks at constrained time cost. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 5353-5360.

He, K., Zhang, X., Ren, S. & Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *In Proceedings of the IEEE international conference on computer vision,* pp. 1026-1034.

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

Hinton, G., Osindero, S. & Teh, Y., 2006. A fast learning algorithm for deep belief nets. *Neural computation,* 18(7), pp. 1527-1554.

Hinton, G., Sejnowski, T. & Poggio, T., 1999. *Unsupervised learning: foundations of neural computation.* eds ed. s.l.:MIT press.

Hirschmuller, H., 2005. Accurate and efficient stereo processing by semi-global matching and mutual information. *In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) ,* Volume 2, pp. 807-814.

Hirschmuller, H., 2006. Stereo vision in structured environments by consistent semi-global matching. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2(06), pp. 2386-2393.

Horn, B. & Schunck, B., 1981. Determining optical flow. *Artificial intelligence,* 17(1-3), pp. 185-203.

Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks,* 4(2), pp. 251-257.

Huang, J., Zou, W., Zhu, Z. & Zhu, J., 2018. An efficient optical flow based motion detection method for non-stationary scenes. *arXiv preprint,* Volume arXiv:1811.08290.

Huang, S., 2010. An advanced motion detection algorithm with video quality analysis for video surveillance systems. *IEEE transactions on circuits and systems for video technology,* 21(1), pp. 1-14.

Huang, S. et al., 2016. Deep learning driven visual path prediction from a single image. *IEEE Transactions on Image Processing,* 25(12), pp. 5892-5904.

Hu, Y. et al., 2018. Adversarial deformation regularization for training image registration neural networks. *n International Conference on Medical Image Computing and Computer-Assisted Intervention,* September, Springer(Cham), pp. 774-782.

Iandola, F. et al., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv,* p. 1602.07360.

Ide, H. & Kurita, T., 2017. Improvement of learning for CNN with ReLU activation by sparse regularization. *In 2017 International Joint Conference on Neural Networks (IJCNN),* May.pp. 2684-2691.

Ilg, E. et al., 2017. Flownet 2.0: Evolution of optical flow estimation with deep networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 2462-2470.

Ince, S. & Konrad, J., 2008. Occlusion-aware optical flow estimation. *IEEE Transactions on Image Processing,* 17(8), pp. 1443-1451.

Ioffe, S. & Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint,* p. arXiv:1502.03167.

Jaderberg, M., Simonyan, K. & Zisserman, A., 2015. Spatial transformer networks. *In Advances in neural information processing systems,* pp. 2017-2025.

Jain, R., Kasturi, R. & Schunck, B., 1995. *Machine vision.* New York: McGraw-Hill.

Jain, R., Kasturi, R. & Schunck, B., 2003. *machine vision.* English Edition ed. New York: McGraw-Hill.

Jammal, S., Tillo, T. & Xiao, J., 2017. Multi-resolution for disparity estimation with convolutional neural networks. *In 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC),* pp. 1756-1761.

Johnson, R. & Zhang, T., 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems,* pp. 315-323.

Kaehler, A. & Bradski, G., 2016. *Learning OpenCV 3: computer vision in C++ with the OpenCV library.* s.l.:" O'Reilly Media, Inc.".

Karpathy, A. & Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 3128-3137.

Karpathy, A. et al., 2014. Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition,* pp. 1725-1732).

Karsch, K., Liu, C. & Kang, S., 2014. Depth transfer: Depth extraction from video using non-parametric sampling. *IEEE transactions on pattern analysis and machine intelligence,* 36(11), pp. 2144-2158.

Keskar, N. & Socher, R., 2017. Improving generalization performance by switching from adam to sgd. Issue arXiv preprint arXiv:1712.07628.

Kim, J. & Kim, T., 2013. Development of Photogrammetric Rectification Method Applying Bayesian Approach for High Quality 3D Contents Production. *Journal of broadcast engineering,* 18(1), pp. 31-42.

Kingma, D. P. & Ba, J. L., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv,* Issue 1412.6980.

Kornysheva, K. & Diedrichsen, J., 2014. Human premotor areas parse sequences into their spatial and temporal features. *Elife,* 3(p.e03043).

Krizhevsky, A., Sutskever, I. & Hinton, G., 2012. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems,* pp. 1097-1105.

Kroeger, T. T. R. D. D. a. V. G. L., 2016. Fast optical flow using dense inverse search. *In European Conference on Computer Vision,* pp. 471-488.

Królak, A. & Strumiłło, P., 2012. Eye-blink detection system for human–computer interaction. *Universal Access in the Information Society,* 11(4), pp. 409-419.

Ladicky, L., Shi, J. & Pollefeys, M., 2014. Pulling things out of perspective. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 89-96.

Laskar, M., Das, A., Talukdar, A. & Sarma, K., 2015. Real-time disparity computation and 3D reprojection for hand gesture recognition. *In 2015 International Symposium on Advanced Computing and Communication (ISACC) ,* pp. 78-83.

LeCun, Y., Bengio, Y. & Hinton, G., 2015. Deep learning. *nature,* 521(7553), p. 436.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 86(11), pp. 2278-2324.

LeCun, Y., Kavukcuoglu, K. & Farabet, C., 2010. Convolutional networks and applications in vision. *In Proceedings of 2010 IEEE International Symposium on Circuits and Systems,* pp. 253-256.

LeCun, Y., Touresky, D., Hinton, G. & Sejnowski, T., 1988. A theoretical framework for back-propagation. *In Proceedings of the 1988 connectionist models summer school,* 1(CMU, Pittsburgh, Pa: Morgan Kaufmann), pp. 21-28.

LeCunY, Y, B. & G, H., 2015. Deep Learning. *Nature,* 521(7553), p. 436.

Lee, W., Lee, E. & Park, K., 2010. Blink detection robust to various facial poses. *Journal of neuroscience methods,* 193(2), pp. 356-372.

Le, V. et al., 2012. Interactive facial feature localization. *In European conference on computer vision,* October.pp. 679-692.

Liang, Z. et al., 2018. Learning for disparity estimation through feature constancy. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2811-2820.

Li, B. et al., 2015. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 1119-1127.

Li, J. et al., 2016. Hierarchical and adaptive phase correlation for precise disparity estimation of UAV images. *IEEE Transactions on Geoscience and Remote Sensing,* 54(12), pp. 7092-7104.

Luo, W., Schwing, A. & Urtasun, R., 2016. Efficient deep learning for stereo matching. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 5695-5703.

Maas, A., Hannun, A. & Ng, A., 2013. Rectifier nonlinearities improve neural network acoustic models. *In Proc. icml,* June, 30(1), p. 3.

Mahapatra, D., Sedai, S. & Garnavi, R., 2018. Elastic registration of medical images with gans. *arXiv preprint arXiv:1805.02369.*

Malik, K. & Smolka, B., 2014. Eye blink detection using local binary patterns. *In 2014 International Conference on Multimedia Computing and Systems (ICMCS),* Volume IEEE, pp. 385-390.

Mao, X. et al., 2017. Least squares generative adversarial networks. *In Proceedings of the IEEE international conference on computer vision,* pp. 2794-2802.

Marcos-Ramiro, A. et al., 2014. Automatic blinking detection towards stress discovery. *In Proceedings of the 16th International Conference on Multimodal Interaction,* Volume ACM, pp. 307-310.

Marsland, S., 2014. *Machine learning: an algorithmic perspective.* s.l.:Chapman and Hall/CRC.

Mayer, N. et al., 2016. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4040-4048.

Medathati, N., Neumann, H., Masson, G. & Kornprobst, P., 2016. Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision. *Computer Vision and Image Understanding,* Volume 150, pp. 1-30.

Mehltretter, M., 2020. Uncertainty Estimation for End-To-End Learned Dense Stereo Matching via Probabilistic Deep Learning. Issue arXiv preprint arXiv:2002.03663.

Meister, S., Hur, J. & Roth, S., 2018. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. *Thirty-Second AAAI Conference on Artificial Intelligence,* Issue April.

Menze, M. & Geiger, A., 2015. Object scene flow for autonomous vehicles. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 3061-3070.

Mikic, I., Krucinski, S. & Thomas, J., 1998. Segmentation and tracking in echocardiographic sequences:Active contours guided by optical flow estimates. *IEEE transactions on medical imaging,* 17(2), pp. 274-284.

Mitrokhin, A., Fermüller, C., Parameshwara, C. & Aloimonos, Y., 2018. Event-based moving object detection and tracking. *In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 1-9.

Mohammed, A., 2014. Efficient eye blink detection method for disabled-helping domain. *Eye,* p. P2.

Moller, M., 1993. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks,* 6(4), pp. 525-533.

Nair, V. & Hinton, G., 2010. Rectified linear units improve restricted boltzmann machines. *In Proceedings of the 27th international conference on machine learning (ICML-10),* pp. 807-814.

Odena, A., Dumoulin, V. & Olah, C., 2016. *Deconvolution and checkerboard artifacts.* [Online] Available at: https://distill.pub/2016/deconv-

checkerboard/?utm_source=mybridge&utm_medium=blog&utm_campaign=read_more [Accessed 9 December 2019].

Oram, D., 2001. Rectification for any epipolar geometry. *In BMVC,* Volume 1, pp. 653-662.

Pan, G., Sun, L., Wu, Z. & Lao, S., 2007. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. *In 2007 IEEE 11th International Conference on Computer Vision ,* pp. 1-8.

Papież, B. et al., 2014. An implicit sliding-motion preserving regularisation via bilateral filtering for deformable image registration. *Medical image analysis,* 18(8), pp. 1299-1311.

Parmar, P. & Chitaliya, M., 2013. Detect eye blink using motion analysis method. *International Journal of Emerging Technologies in Computational and Applied Sciences(IJETCAS).*

Portello, J., Rosenfield, M. & Chu, C., 2013. Blink rate, incomplete blinks and computer vision syndrome. *Optometry and Vision Science,* 90(5), pp. 482-487.

Press, W. & Teukolsky, S., 1990. Savitzky-Golay smoothing filters. *Computers in Physics,* 4(6), pp. 669-672.

Qin, C. et al., 2019. Unsupervised deformable registration for multi-modal images via disentangled representations. *In International Conference on Information Processing in Medical Imaging,* June, Springer(Cham), pp. 249-261.

Rahman, A., Sirshar, M. & Khan, A., 2015 . Real time drowsiness detection using eye blink monitoring. *IEEE National Software Engineering Conference (NSEC) ,* pp. 1-7.

Raichle, M., 2010. Two views of brain function. *Trends in cognitive sciences,* 14(4), pp. 180-190.

Ranjan, A. & Black, M., 2017. Optical flow estimation using a spatial pyramid network. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4161-4170.

Revaud, J., Weinzaepfel, P., Harchaoui, Z. & Schmid, C., 2016. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision,* 120(3), pp. 300-323.

Rezaei, M. & Klette, R., 2012. Novel adaptive eye detection and tracking for challenging lighting conditions. *In Asian Conference on Computer Vision,* pp. 427-440.

Roberts, L., 1963. *Machine Perception of Three-Dimensional Solids,* s.l.: Massachusetts Institute of Technology.

Roberts, R., Potthast, C. & Dellaert, F., 2009. Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies. June, pp. 57-64.

Rosu, M. & Hugo, G., 2012. Advances in 4D radiation therapy for managing respiration: part II–4D treatment planning. *Zeitschrift für Medizinische Physik,* 22(4), pp. 272-280..

Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv.*

Rumelhart, D., Hinton, G. & Williams, R., 1986. Learning representations by back-propagating errors. *nature,* 323(6088), pp. 533-536.

Samer, J., T, T. & X, J., 2017. multi-resolution for disparity estimation with convolutional neural network. *Proceeding of APSIPA Annual Summit and Conference,* pp. 1756-1761.

Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A., 2018. How does batch normalization help optimization?. *In Advances in Neural Information Processing Systems,* pp. 2483-2493.

San, T. & War, N., 2017. Feature based disparity estimation using hill-climbing algorithm. *IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA),* pp. 129-133.

Saragih, J., Lucey, S. & Cohn, J., 2011. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision,* 91(2), pp. 200-215.

Savva, M., Chang, A. & Hanrahan, P., 2015. Semantically-enriched 3D models for common-sense knowledge. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops,* pp. 24-31.

Saxena, A., Sun, M. & Ng, A., 2007. Learning 3-d scene structure from a single still image. *In 2007 IEEE 11th International Conference on Computer Vision,* pp. 1-8.

Scharstein, D. et al., 2014. High-resolution stereo datasets with subpixel-accurate ground truth. *In German conference on pattern recognition,* September, Issue Springer, C, pp. 31-42.

Sentker, T., Madesta, F. & Werner, R., 2018. GDL-FIRE $$^\text {4D} $$: Deep Learning-Based Fast 4D CT Image Registration. *In International Conference on Medical Image Computing and Computer-Assisted Intervention,* September., Issue Springer, Cham, pp. 765-773.

Shi, W. et al., 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 1874-1883.

Simonyan, K. & Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv,* p. 1409.1556.

Song, F., Tan, X., Liu, X. & Chen, S., 2014. Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients. *Pattern Recognition,* 47(9), pp. 2825-2838.

Soukupova, T. & Cech, J., 2016. *May. Eye blink detection using facial landmarks.* Rimske Toplice, Slovenia., In 21st Computer Vision Winter Workshop.

Srivastava, R., Greff, K. & Schmidhuber, J., 2015. Highway networks. *arXiv preprint arXiv:1505.00387.*

Sugawara, Y., Shiota, S. & Kiya, H., 2019. Checkerboard artifacts free convolutional neural networks. *APSIPA Transactions on Signal and Information Processing,* Volume 8.

Suhr, J., 2009. Kanade-lucas-tomasi (klt) feature tracker. *Computer Vision (EEE6503),* pp. 9-18.

Sundaram, N., Brox, T. & Keutzer, K., 2010. Dense point trajectories by GPU-accelerated large displacement optical flow. *European conference on computer vision,* Issue Springer, pp. 438-451.

Sun, Y., Zafeiriou, S. & Pantic, M., 2013. A hybrid system for on-line blink detection. *In Hawaii International Conference on System Sciences.*

Szegedy, C. L. W. J. Y. S. P. et al., 2015. Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 1-9.

Szeliski, R., 2011. *Computer vision: algorithms and applications.* Washington: Springer Science & Business Media.

Szwoch, M. & Pieniążek, P., 2012. Eye blink based detection of liveness in biometric authentication systems using conditional random fields. *In International Conference on Computer Vision and Graphics,* pp. 669-676).

Tan, K., Kriegman, D. & Ahuja, N., 2002. Appearance-based eye gaze estimation. *In Sixth IEEE Workshop on Applications of Computer Vision, 2002,* Volume (WACV 2002), pp. 191-195.

Tanner, C. et al., 2018. Generative adversarial networks for MR-CT deformable image registration. *arXiv preprint arXiv,* p. 1807.07349.

Tao, T., Koo, J. & Choi, H., 2008. A fast block matching algorthim for stereo correspondence. *IEEE Conference on Cybernetics and Intelligent Systems,* pp. 38-41.

Tato, A. & Nkambou, R., 2018. Improving adam optimizer.

Tekalp, M. M., 2015. *Digital Video Processing.* 2nd ed. Massachussetts: Prentice Hall Signal Processing Series.

Tenbrinck, D. et al., 2013. Histogram-based optical flow for motion estimation in ultrasound imaging. *Journal of Mathematical Imaging and Vision,* 47(1-2), pp. 138-150.

Theodoridis, S., 2015. *Machine learning: a Bayesian and optimization perspective.* s.l.:Academic Press.

Tietjen, G. & Moore, R., 1972. Some Grubbs-type statistics for the detection of several outliers. *Technometrics,* 14(3), pp. 583-597.

Tomasi, C. & Kanade, T., 1991. Detection and Tracking of points features. *Computer Science Department, Carnege Mellon University, Tech. rep.*

Tonioni, A., Poggi, M., Mattoccia, S. & Di Stefano, L., 2017. Unsupervised adaptation for deep stereo. *In Proceedings of the IEEE International Conference on Computer Vision,* pp. 1605-1613.

Ullrich, K., Meeds, E. & Welling, M., 2017. Soft weight-sharing for neural network compression. *arXiv,* Issue 1702.04008.

Vandemeulebroucke, J. et al., 2012. Automated segmentation of a motion mask to preserve sliding motion in deformable registration of thoracic CT. *Medical physics,* 39(2), pp. 1006-1015.

Viola, P. & Jones, M., 2004. Robust real-time face detection. *International journal of computer vision,* 57(2), pp. 137-154.

Waibel, A. et al., 1995. Phoneme recognition using time-delay neural networks. *Theory, Architectures and Applications,* pp. 35-61.

Wang, Y. et al., 2019. Assessing Optimizer Impact on DNN Model Sensitivity to Adversarial Examples. *IEEE Access,* Volume 7, pp. 152766-152776.

Wang, Y., Ostermann, J. & Zhang, Y., 2002. *Video Processing & Communications.* New Jersey: Prentice-Hall, Inc.

Wei, S., Yang, L., Chen, Z. & Liu, Z., 2011. Motion detection based on optical flow and self-adaptive threshold segmentation. *Procedia Engineering,* Volume 15, pp. 3471-3476.

Wu, Y. et al., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXi.*

Yamamoto, T. et al., 2016. The first patient treatment of computed tomography ventilation functional image-guided radiotherapy for lung cancer. *Radiotherapy and Oncology,* 118(2), pp. 227-231..

Yang, X., Kwitt, R., Styner, M. & Niethammer, M., 2017. Quicksilver: Fast predictive image registration–a deep learning approach. *NeuroImage,* Volume 158, pp. 378-396.

Yan, P., Xu, S., Rastinehad, A. & Wood, B., 2018. Adversarial image registration with application for MR and TRUS image fusion. *In International Workshop on Machine Learning in Medical Imaging,* September, Springer(Cham), pp. 197-204.

Yao Wang, Y.-q. Z.-q. Z., 2001. Video Processing and Communications. In: New Jersey: Prentice Hall, pp. 114-116.

Yin, Z. & Shi, J., 2018. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1983-1992.

Yong, X., 2013. Improved Gaussian Mixture Model in Video Motion Detection. *Journal of Multimedia,* 8(5).

Zach, C. P. T. a. B. H., 2007. A duality based approach for realtime TV-L 1 optical flow. *In Joint pattern recognition symposium,* pp. 214-223.

Zagoruyko, S. & Komodakis, N., 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146.*

Zbontar, J. & LeCun, Y., 2016. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *Journal of Machine Learning Research,* 17(1-32), p. 2.

Zhang, C. & Woodland, P., 2015. Parameterised sigmoid and ReLU hidden activation functions for DNN acoustic modelling. *In Sixteenth Annual Conference of the International Speech Communication Association.*

Zhang, J. & Hu, J., 2008. Image segmentation based on 2D Otsu method with histogram analysis. *In 2008 International Conference on Computer Science and Software Engineering,* Volume 6, pp. 105-108.

Zhang, Q., Yang, L., Chen, Z. & Li, P., 2018. A survey on deep learning for big data. *Information Fusion,* Volume 42, pp. 146-157.

Zhang, Y., Chan, W. & Jaitly, N., 2017. Very deep convolutional networks for end-to-end speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) ,* pp. 4845-4849.

Zhang, Y. et al., 2020. Deeper insights into weight sharing in neural architecture search. p. arXiv preprint arXiv:2001.01431.

Zhang, Z., 2012. Camera parameters (intrinsic, extrinsic. *Computer Vision: A Reference Guide,* pp. 81-85.

Zhou, J. X. W. & Chellali, R., 2017. Analysing the effects of pooling combinations on invariance to position and deformation in convolutional neural networks. *In IEEE International Conference on Cyborg and Bionic Systems (CBS),* pp. 226-230.