

A DEEP LEARNING APPROACH TO SURFACE RECONSTRUCTION FOR
SURGICAL NAVIGATION DURING LAPAROSCOPIC, ENDOSCOPIC OR
ROBOTIC SURGERY



A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
MIDDLE EAST TECHNICAL UNIVERSITY
BY

AMIN ZABARDAST

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF MEDICAL INFORMATICS

AUGUST 2019

Approval of the thesis:

**A DEEP LEARNING APPROACH TO SURFACE RECONSTRUCTION FOR
SURGICAL NAVIGATION DURING LAPAROSCOPIC, ENDOSCOPIC OR
ROBOTIC SURGERY**

Submitted by **AMIN ZABARDAST** in partial fulfillment of the requirements for the degree of **Master of Science in Medical Informatics Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, Graduate School of **Informatics**

Assoc. Prof. Dr. Yeşim Aydın Son
Head of Department, **Health Informatics, METU**

Prof. Dr. Ünal Erkan Mumcuoğlu
Supervisor, **Health Informatics, METU**

Examining Committee Members:

Assist. Prof. Dr. Aybar Can Acar
Health Informatics, METU

Prof. Dr. Ünal Erkan Mumcuoğlu
Health Informatics, METU

Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu
Multimedia Informatics, METU

Assist. Prof. Dr. Tolga İnan
Electrical and Electronics Engineering, Çankaya University

Assoc. Prof. Dr. Nurcan Tunçbağ
Health Informatics, METU

Date:



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: AMIN ZABARDAST

Signature :

ABSTRACT

A DEEP LEARNING APPROACH TO SURFACE RECONSTRUCTION FOR SURGICAL NAVIGATION DURING LAPAROSCOPIC, ENDOSCOPIC OR ROBOTIC SURGERY

Zabardast, Amin

M.S., Department of Medical Informatics

Supervisor : Prof. Dr. Ünal Erkan Mumcuoğlu

August 2019, 102 pages

Minimally invasive surgical procedures utilize technology to provide surgeons with more functionality as well as a better perspective to help them succeed in their tasks and reduce operations risks. Surgeons usually rely on screens and cameras during minimally invasive surgeries such as Laparoscopic, Endoscopic, or Robotic Surgeries. Currently, operating rooms use information from different modalities such as Computer-Aided Tomography and Magnetic Resonance Imaging. However, the information is not integrated, and the task of extracting and combining features falls under the surgeon's expertise. Conventional cameras, although very helpful, are not capable of transmitting every aspect of the scene including depth perception. Recently stereo cameras are being introduced to operating rooms. Utilizing stereo endoscopic equipment alongside algorithms to process the information can enable depth perception. The process of extracting depth information from stereo cameras, also known as Stereo Correspondence, is still an active research field in computer science. Understanding depth information from the view is a necessary step for reconstruction of the scene in a 3D environment. Ultimately, this reconstructed environment acts as a basis to build an Augmented Reality with extra information baked into the scene to help the surgeon. Artificial Neural Networks (ANNs), specially Convolutional Neural Networks (CNNs), have revolutionized the computer vision research in the past few years. One of the problems that researchers tried to solve using ANNs was Stereo Correspondence. There are variations of CNNs with excellent accuracy in Stereo Correspondence problem. This thesis aims to achieve surface reconstruction from in vitro stereo images of organs using Deep Neural Networks and in silico simulations.

Keywords: Convolutional Neural Network, Surface Reconstruction, Stereo Imaging, Surgical Navigation, Augmented Reality



ÖZ

LAPAROSKOPIK, ENDOSKOPIK VE ROBOTİK CERRAHİDE NAVİGASYON (YÖN BULMA) AMACIYLA DERİN ÖĞRENME YAKLAŞIMIYLA ORGAN YÜZEYİ OLUŞTURMA

Zabardast, Amin

Yüksek Lisans, Tıp Bilişimi Bölümü

Tez Yöneticisi : Prof. Dr. Ünal Erkan Mumcuoğlu

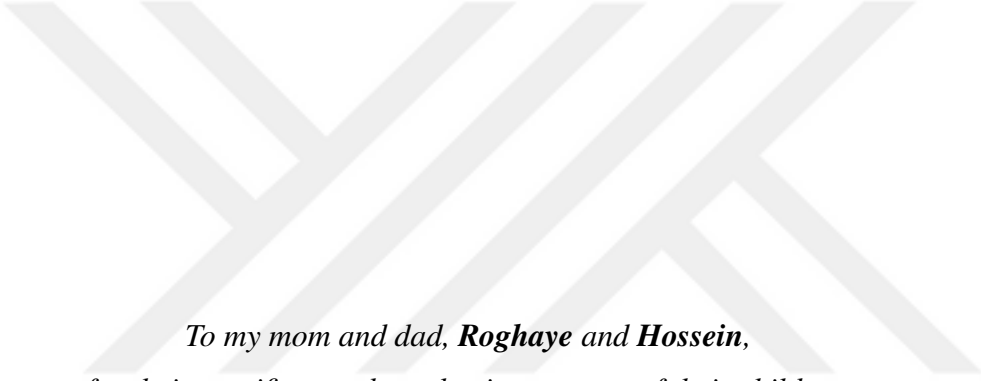
Ağustos 2019 , 102 sayfa

En-az-girişimsel cerrahi yöntemler cerrahlara uyguladıkları operasyonlarda yüksek başarı sağlamalarına, olası riskleri azaltarak, onlara işlevsel araçlar kazandırma ve daha iyi bir perpektif sağlayan teknolojik imkanlar sunarlar. Cerrahlar en-az-girişimsel cerrahi operasyonlar sırasında (Laparoskopik, Endoskopik veya Robotik cerrahi) genellikle ekrandaki görüntüleme ve kameralara bağlı çalışırlar. Ameliyathanelerde farklı görüntüleme kiplerinden gelen bilgi (görüntüler) kullanılır (Bilgisayarlı Tomografi, Manyetik Resonans gibi). Ancak, bu bilgiler kamera görüntüleriyle bütünleştirilmez ve bu bilginin çıkartılarak birleştirilmesi ancak cerrahın deneyim ve yetenekleriyle sınırlı kalır. Klasik kameralar, çok faydalı olmakla birlikte, olayın her yönünü (mesela derinlik bilgisi) iletmekte yeterli olmazlar. Son yıllarda, steryo kameralar ameliyathanelerde kullanıma sunulmuştur. Steryo endoskopik araçlar ve beraberindeki algoritmalar derinlik algılamasına yardımcı olurlar. Derinlik bilgisinin steryo kameralardan çıkarımını sağlayan yöntemler ('steryo uyuşma' olarak bilinir) halen bilgisayar bilimleri alanında aktif bir araştırma alanıdır. 3 boyutlu ortamın sayısal oluşturulması için gerekli basamak derinlik bilgisidir. Bu bilgi, cerraha yardımcı olacak olan 'eklenmiş gerçeklik' ortamı oluşturmak için temel teşkil eder. Yapay Sinir Ağları (YSA) ve özellikle de Evrişimsel Sinir Ağları (ESA) son yıllarda bilgisayarlı görü alanındaki araştırmalarda çığır açmıştır. Araştırmacıların YSA kullanarak çözmeye çalıştığı konuların arasında 'steryo uyuşma' da vardır. Steryo uyuşma konusunda ESA'nın değişik türleri oldukça iyi başarımlar göstermiştir. Bu tez çalışması, derin sinir ağları ve bilgisayar benzetimleri kullanarak, hayvan organ dokularından alınmış steryo görün-

tülerinde yüzey geri-çatım problemini hedeflemiştir.

Anahtar Kelimeler: Evrişimsel Sinir Ağları, Yüzey Geri-Çatımı, Steryo Görüntüleme, Cerrahi Yönlendirme (Navigasyon), Eklenmiş Gerçeklik





*To my mom and dad, **Roghaye** and **Hossein**,
for their sacrifices and everlasting support of their children.*

*To my hardworking sister **Sara**, who always reminds me of dedication.*

*To my lively brother **Ehsan**, whose attitude towards living is admirable.*

*To my lovely wife **Mina**, whose different view on life has inspired me for years.*

ACKNOWLEDGMENTS

I would like to express my appreciation to my advisor, Prof. Dr. Erkan Mumcuođlu, for his invaluable and constructive suggestions during the planning and development of this research work. This research would have not been possible without his willingness to generously give his time and advice.

I would like to thank Asst. Prof. Dr. Aybar Acar for granting the access to a powerful computational platform, without which training the Deep Neural Networks would not be feasible.

I would also like to extend my thanks to Assoc. Prof. Dr. Yeřim Aydın Son for her help during my applications and registration to the Graduate School of Informatics in Middle East Technical University. I would not be able to start the school after arriving late to Ankara, if it was not for her help before and during the process of preregistration.

I would like to thank Assoc. Prof. Dr. Nurcan Tunçbađ, Assoc. Prof. Dr. Hüseyin Hacıhabibođlu, and Assist. Prof. Dr. Tolga İnan for taking time out from their busy schedules to attend the examining committee and provide insightful comments.

I am grateful to all of my friends, Ali Torabi, Yeřim Semchenko, and everyone else, for all their moral and emotional support throughout my studies.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	vi
ACKNOWLEDGMENTS	ix
LIST OF ABBREVIATIONS	xix
CHAPTERS	
1 INTRODUCTION	1
1.1 Motivation	2
1.1.1 Stereo Imaging in Medical Field	3
1.1.2 Stereo Matching and Machine Learning	3
1.2 Objectives of this Thesis	3
1.3 Outline of this Thesis	5
2 STEREO VISION	7
2.1 Steps in the stereo vision process	8
2.1.1 Image Acquisition	8
2.1.2 System Geometry	9
2.1.3 Feature Extraction	9
2.1.4 The Matching Problem	9
2.1.5 Depth Calculation	9
2.1.6 Interpolation	9
2.2 Stereo Correspondence Problem	10
2.3 Epipolar Geometry	12
2.3.1 Important Questions	12

2.3.2	Epipolar Restriction	13
2.4	Algebraic representation of epipolar geometry	14
2.4.1	The fundamental matrix F	14
2.4.2	Estimation of fundamental matrix F	14
2.5	Image Rectification	15
2.6	Matching Algorithms	16
2.6.1	Block Matching	17
2.6.2	Block Matching with Adaptive Window	18
2.6.3	Feature Matching	18
2.6.4	Dynamic Programming	19
2.6.5	Intrinsic Curves	20
2.6.6	Graph Cuts	20
2.6.7	Belief Propagation	22
2.7	Summary	23
3	MACHINE LEARNING	25
3.1	Representation Learning	26
3.2	Deep Learning	27
3.3	Defining a Learning Algorithm	28
3.3.1	The Task T	28
3.3.2	The Performance Measure P	29
3.3.3	The Experience E	31
3.4	Capacity of a Learning Algorithm	32
3.4.1	Training and Testing Errors	32
3.4.2	Overfitting and Underfitting	32
3.5	Supervised Learning Algorithms	33
3.6	Unsupervised Learning Algorithms	37
3.7	Gradient Descent	39
3.7.1	Gradient-Based Optimization	39
3.7.2	Stochastic Gradient Descent	40
3.8	Summary	41
4	PROBLEM, DATA, AND SIMULATION	43
4.1	Why Is Data Important?	44

4.1.1	What Makes A Good Dataset For Deep Learning?	44
4.2	The Problem	45
4.3	Exploring The Problem’s Domain	47
4.4	Existing Datasets	51
4.4.1	TMI Dataset	52
4.5	Why In-Silico Simulation?	54
4.5.1	Related Research	54
4.5.2	A Hypothesis	55
4.6	<i>In-Silico</i> Simulation	55
4.6.1	Simulation Environment	55
4.6.2	Simulation Parameters	56
4.6.3	Dataset Parameters	61
4.7	Summary	62
5	A DEEP LEARNING SOLUTION, AND RESULTS	63
5.1	Deep Feed-Forward Networks	64
5.1.1	Computational Graph	64
5.1.2	Back Propagation	66
5.1.3	Convolutional Neural Networks	66
5.1.4	Auto-Encoders	69
5.2	The Solutions and Training Process	71
5.2.1	Type S Network: The Simple Solution	71
5.2.2	Type C Network: The Correlational Layer Solution	71
5.2.3	Type CS Network: The Ensemble Solution	74
5.2.4	Network Input Size Issue	74
5.2.5	Training Parameters	74
5.2.6	Training Process and Performance	76
5.3	Evaluation and Results	79
5.3.1	Comparative Study and Control Methods	80
5.3.2	Validation Set Results	81
5.3.3	Back Projection To 3D Space	81
5.3.4	TMI Dataset Results	82
5.3.5	Type CS Network’s Generalization Problem	89

5.4 Summary	90
6 CONCLUSION	91
6.1 Summary Of Results	92
6.2 Discussion	92
6.3 Future Studies	93
REFERENCES	95



LIST OF TABLES

TABLES

Table 1.1:	CNN based methods in KITTI 2015 leader board as of July 2019	4
Table 2.1:	Summary of fundamental matrix properties	14
Table 2.2:	Stereo Matching Methods	17
Table 4.1:	Openly available surgical endoscopic datasets	52
Table 4.2:	The partitions that divide the dataset into clusters of similar images	62
Table 5.1:	Network Type S structure guide	72
Table 5.2:	Network Type C structure guide	73
Table 5.3:	The metrics used in this research.	78

LIST OF FIGURES

FIGURES

Figure 1.1: FOV examples in Laparoscopic Surgery	4
Figure 2.1: Displacement of the representations of a point in stereo vision . . .	8
Figure 2.2: Steps Required for Stereo Vision Process	8
Figure 2.3: Disparity/depth map for a scene	10
Figure 2.4: Point correspondence geometry	11
Figure 2.5: Epipolar Geometry	13
Figure 2.6: Image rectification setup	17
Figure 2.7: Block matching on two rectified images	18
Figure 2.8: Disparity created using dynamic programming	19
Figure 2.9: Disparity Space Image	20
Figure 2.10: A representation of max-flow/min-cut setup in stereo matching problem	21
Figure 2.11: Stereo modeled with MRF	22
Figure 2.12: Message Passing in Belief Propagation	23
Figure 3.1: Relation of Deep Learning to Machine Learning and AI	26
Figure 3.2: Building representations in a hierarchy by deep neural networks . .	27
Figure 3.3: Confusion matrix in binary classification	31
Figure 3.4: Examples of underfitting and overfitting	33
Figure 3.5: Training Error and Testing Error in different fitting scenarios . . .	33
Figure 3.6: The regression line calculated by Least Mean Squared algorithm .	34
Figure 3.7: Comparing Linear Regression and Logistic Regression's perfor- mance with categorical data	35
Figure 3.8: An example of a decision tree.	35

Figure 3.9: Visual Comparison of Decision Tree and Random Forest Results	36
Figure 3.10: Visualizing the support vectors and separation plane created by SVM	37
Figure 3.11: A dataset described by the mixture of two Gaussian models	38
Figure 3.12: A dataset divided into 4 categories using <i>k</i> -means algorithm	39
Figure 4.1: Minimally Invasive Surgery	46
Figure 4.2: Augmented Reality in Minimally Invasive Surgery	46
Figure 4.3: Path to achieving Augmented Reality in MIS	47
Figure 4.4: Occlusion of the surface by surgical tools	47
Figure 4.5: Featureless tissues present in the view	48
Figure 4.6: Reflections from wet organ surface in MIS.	48
Figure 4.7: Non matching reflections on the surface.	48
Figure 4.8: Insufficient lighting in MIS	49
Figure 4.9: Smoke present in the surgical environment	49
Figure 4.10: Examples of blood on organ surface in MIS	50
Figure 4.11: Use of water during MIS operation	50
Figure 4.12: Example of lens distortion on images	51
Figure 4.13: Motion blur visible in surgical footage in MIS	51
Figure 4.14: Examples from TMI dataset	53
Figure 4.15: Image acquisition process of TMI dataset	53
Figure 4.16: Blender three-dimensional computer graphics software as the sim- ulation environment	56
Figure 4.17: Disparity, calculated by Semi-Global Block Matching, and its his- togram.	57
Figure 4.18: An example texture used in the study	58
Figure 4.19: Histogram for all of the TMI dataset disparities.	59
Figure 4.20: Smoke examples from TMI dataset.	60
Figure 4.21: Examples from simulations.	61
Figure 5.1: Non-linearity (activation) functions commonly used in neural net- works.	64
Figure 5.2: An example computational graph for $a^2 + 2ab + b^2$	65
Figure 5.3: Feed Forward and Back Propagation on a simple Neural Network.	66
Figure 5.4: Example of Convolution Operator in a Convolutional Layer.	67

Figure 5.5: Max Pooling layer reduces the information in order to extract better features.	68
Figure 5.6: Example of a convolutional neural network structure	68
Figure 5.7: General Structure of an Auto-Encoder	69
Figure 5.8: An undercomplete Autoencoder	70
Figure 5.9: The structure of the FlowNet and it's correlational layer	72
Figure 5.10: The divisions over a TMI dataset image.	75
Figure 5.11: The shape of Log-Cosh Loss Function	75
Figure 5.12: The loss per epoch for the training dataset	77
Figure 5.13: The loss per epoch for the validation dataset	77
Figure 5.14: The performance on training dataset, compared using different metrics	78
Figure 5.15: The performance on validation dataset, compared using different metrics	79
Figure 5.16: The validation tool available for evaluating the results for TMI dataset	80
Figure 5.17: The box plots comparing the performance of the different methods.	82
Figure 5.18: The RMS distance (mm) of all the images withing the TMI dataset with their ground truth.	84
Figure 5.19: The RMS distance (mm) of all the images with smoke in view, withing the TMI dataset with their ground truth.	84
Figure 5.20: The RMS distance (mm) of all the images with smoke in view, withing the TMI dataset with their ground truth.	85
Figure 5.21: The RMS distance (mm) of all the images with blood in view, withing the TMI dataset with their ground truth.	85
Figure 5.22: The RMS distance (mm) of all the images with blood in view, withing the TMI dataset with their ground truth.	86
Figure 5.23: The RMS distance (mm) of all the images with direct and angular views, withing the TMI dataset with their ground truth.	86
Figure 5.24: The RMS distance (mm) of all the images with different distances, with their ground truth.	87
Figure 5.25: Comparing In-Vitro Reconstructions by Type C network (left) and Semi-Global Matching (right)	88

Figure 5.26: Comparing In-Vitro Reconstructions of a target scene with smoke
(right) and without smoke (left) 89



LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
AO	Ambient Occlusion
AR	Augmented Reality
BM	Block Matching
BP	Belief Propagation
CNN	Convolutional Neural Network
CSV	Comma Separated Values
CT	Computed Tomography
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DSI	Disparity Space Image
FOV	Field of View
GMM	Gaussian Mixture Model
IGS	Image Guided Surgery
LiDAR	Light Detection and Ranging
LMS	Least Mean Squared
MAP	Maximum A Posteriori
MIS	Minimally Invasive Surgery
ML	Machine Learning
MLP	Multilayered Perceptron
MR	Magnetic Resonance
MRF	Markov Random Field
MSE	Mean Squared Error
NCC	Normalized Cross-Correlation
NSSD	Normalized Sum of Squared Distances
OCR	Optical Character Reconstruction
PFM	Probability Mass Function
PFM	Portable Float Map
PNG	Portable Network Graphics
ReLU	Rectified Linear Unit

RMS	Root Mean Square
SAD	Sum of Absolute Distances
SGD	Stochastic Gradient Descent
SGM	Semi-Global Matching
SL	Structured Light
SSD	Sum of Squared Distances
SVM	Support Vector Machine
TOF	Time of Flight
ZNCC	Zero Normalized Cross-Correlation



CHAPTER 1

INTRODUCTION

Computers have been a part of our daily lives for decades, and they are integrated into all branches of science and technology. Nowadays, life without computers seems like an impossible task. Moreover, all the fields and professions rely on computers, whether directly or indirectly. The same is true about the applications of Computer Vision (CV) and Artificial Intelligence (AI). This becomes more prominent when we understand how much of the products we daily use utilize some amount of AI and CV to function.

Machine Learning (ML) is an important subfield of AI that learns to do a range of different tasks by experiencing either some representations of the data about the task or the raw data itself. ML is a powerful tool that has enabled the automation of many tasks that seemed to be impossible to code decades ago. We rely on the ML algorithm on a daily basis without realizing it. For example, weather forecast, suggestions we receive about products we may be interested in, our personalized search result on the internet, and the spam filter in our emails are all some example of the influence of the ML.

Deep Learning (DL) algorithms are a specific type of ML algorithms that are designed to solve some intuitive tasks like object speech recognition that is easy for humans to do but hard to describe using formal language [29]. In the past decade, we witnessed a huge rise in research interest around this subject. Researchers are trying to solve many complex problems using DL, and there are many successful examples of these DL applications. For example, speech recognition, which was considered to be a difficult problem to solve, is now integrated into our lives via mobile devices. Also, autonomous drive vehicles have become a reality, and they utilize a lot of deep learning in their systems.

The medical domain is also being affected by the rise of ML and DL. Expert systems like IBM Watson are providing the doctors with second opinions on their patients. DL also has been used to detect early stages of many illnesses like cancer and consequently improved the quality of life for these patients alongside their longevity. There are many expert systems that even do not rely on processed data and can work with a patient's raw data, like Computed Tomography (CT) and Magnetic Resonance (MR) images.

This study aims to enable Augmented Reality (AR) based surgical navigation systems by creating a three-dimensional visual representation of the surgical environment us-

ing DL. To achieve this, we should solve the Stereo Correspondence problem and estimate the depth information.

1.1 Motivation

Depth estimation has been a highly active research field in computer vision for the past few decades with numerous applications such as, but not limited to, object tracking and detection [20], robot navigation [63], autonomous navigation and obstacle detection for auto driving cars [38], minimally invasive surgery [79, 55], three-dimensional surface reconstruction [68] and so on.

Currently, there are four main depth estimation methods:

- LiDAR
- Time Of Flight (TOF) Cameras
- Structured Light (SL) Scanning
- Stereo Cameras

LiDAR uses laser technology to estimate depth information. This makes LiDAR an accurate depth measurement method. Although there are a lot of advances in LiDAR technology, this method still has considerable limitations. The main issue with using LiDAR is that its equipment is bulky and expensive, which in turn making it less practical and less accessible for common use. TOF cameras use an integrated light source to create pluses which illuminates the scene and by observing the reflected light, the distance estimation, or depth information, can be calculated. These cameras are specifically designed for depth estimation, and there is no need for heavy digital image processing while using them. However, the accuracy of estimation will become much lower when the distance between object and camera increases [88, 41]. The resolution of images provided by TOF camera is low [47, 41], and multiple reflections from objects convex corners affects the camera's functionality. Additionally, ambient light (like and outdoors areas or a well illuminated operating room) will disrupt TOF camera's ability to calculate depth information. SL scanning method uses projections of known patterns of light, to calculate the depth estimation of a scene. SL scanning uses structured patterns of light (usually grid, horizontal or vertical stripes) to illuminate the scene. Then by capturing the scene with two other cameras from different angles, the three-dimensional scene can be reconstructed. SL cameras have many applications from entertainment to robotic assembly. However, they also have some limitations. SL scanning has weak performance in a setting with ambient light or if the scene has some distance from the projection source. Also, this method has a low performance in the case of objects with a surface that reflects the light [65].

Many stereo matching algorithms have been developed with the purpose of calculating high-quality disparity map using only a stereo camera setup. Although the price of this equipment is low, the computational power required to estimate the depth information is high, and to this day, disparity calculation remains a resource

consuming computation. Moreover, computation complexity increases when using high-resolution images or videos.

Despite heavy computational requirements, extracting depth information using stereo cameras is an active branch of computer vision research. The convenience of the equipment and some other factors such as relying solely on visible light makes extracting disparity map from stereo cameras an interesting topic for researchers. Many new algorithms are developed and benchmarked on standardized datasets like Middlebury [84] and KITTI [27, 57, 58].

1.1.1 Stereo Imaging in Medical Field

Stereo Imaging techniques are being used in the medical field from as early as the 2000s, but research interest in using this type of imaging for medical purposes has increased in recent years. This is because of advances in computation technology and increase in the convenience of access. Use of Stereo Imaging can lead to an improvement in surgical accuracy, increase in patient safety, and reducing operation time [64]. Some applications of Stereo Imaging in the medical field are laparoscopic surgery [11], surgical navigation, and robotic-assisted performances [62].

1.1.2 Stereo Matching and Machine Learning

Use of Machine Learning (ML) and Deep Learning (DL) to solve the stereo matching problem is a relatively novel approach. Solving this problem with Convolutional Neural Networks (CNN) gained attention after an article by Žbontar and Le Cun [98], published in 2015. Authors of this article created a successful method to calculate the matching cost of a stereo algorithm using Artificial Neural Network (ANN) and reported that their method was the top performing method in KITTI dataset at August of 2014 [98]. Since 2015, more articles used CNN for disparity matching and currently (July 2019) many CNN based methods are amongst the top performing methods in Middlebury and KITTI leader boards some of which are mentioned in Table 1.1.

1.2 Objectives of this Thesis

Medical related environments have the potential to be drastically different than everyday scenery. This specificity requires specially designed algorithms, tuned to a specific problem. An example is the use of stereo cameras in laparoscopic surgery. Extracting depth information from the environment inside of a patient's abdomen is challenging in several ways. Firstly, the tissue on organs may have a homogeneous structure which will make stereo matching difficult. Additionally, a certain level of robustness is required for scenes with complex illumination or the presence of blood or smoke in the field of view (FoV). Moreover, occlusions commonly happen by surgical instruments [55] as viewed in Figure 1.1c.

Table 1.1: CNN based methods among top 15 entries in KITTI 2015 [57, 58] leader board as of July 2019.

Rank	Method	D1-all ¹	Runtime
1	M2S_CSPN [17]	1.74 %	0.5 s
2	GANet-deep[100]	1.81 %	1.8 s
3	AMNet [23]	1.84 %	0.9 s
4	AcfNet	1.89 %	0.48 s
6	RawStereoNet [42]	1.9 %	0.43 s
7	ASNet_s	1.93 %	1.5 s
8	MS_CSPN [17]	1.93 %	0.5 s
9	GANet-15 [100]	1.93 %	0.36 s
10	NCA-Net	1.94 %	0.5 s
11	APMNet	1.95 %	0.5 s
12	ASONet	1.97 %	1.5 s
13	PSMNet_R [16]	1.98 %	0.5 s
14	ASNet_t	2.0 %	1.5 s
15	HD ³ -Stereo [97]	2.2 %	0.14 s

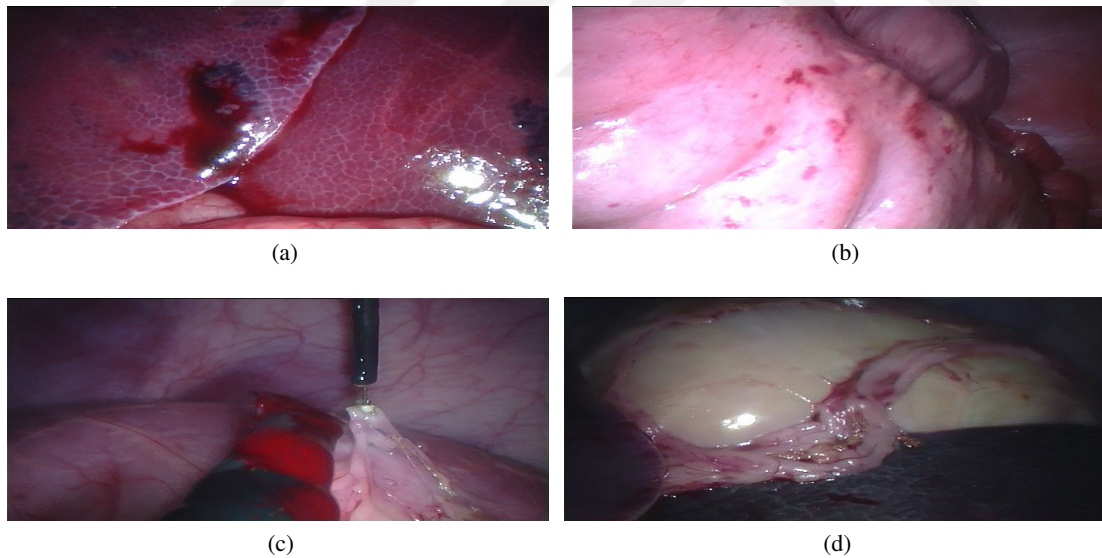


Figure 1.1: FOV examples in Laparoscopic Surgery.

The primary objective of this thesis is to study benefits and challenges of using DL based algorithms for extracting depth information from Medical/Surgical environments and attempting to create Deep Neural Networks (DNN) to study this problem. A secondary objective is to provide an open dataset to use in training ML algorithms for medical operations. To the best of our knowledge, no one has attempted a similar study before.

¹ Percentage of stereo disparity outliers in first frame

1.3 Outline of this Thesis

The remainder of this thesis is divided into five chapters. In Chapter 2 basics of Stereo Camera Models and Stereo Correspondence such as Epipolar Geometry, Camera Calibration, and Rectification will be presented. Chapter 3 will cover the basics of Machine Learning (ML), and some definitions/examples to familiarize the reader about the topic. In Chapter 4, we analyze the problem domain and its characteristics that make it difficult to work with images from this domain. We also look at existing datasets, their advantages, and their problems. Then we explain the hypothesis and the reason for attempting simulation. The simulation characteristics and parameters are also explained in this section. In Chapter 5, we briefly review the concepts behind Deep Neural Networks and their theory. Afterwards, we explain our methodology in solving the Stereo Correspondence problem using deep learning and report the result of experiments on both synthesized data and real data. Finally, conclusion, discussion, and future research possibilities are presented in Chapter 6.



CHAPTER 2

STEREO VISION

Computer vision is arguably amongst one of the most studied computer science subjects. In the classic approach to computer vision, a single camera had been used as a sensor to obtain information from the environment [80]. However, an advancement to the field happened when multiple cameras were introduced to collect information from the environment [24]. Most of the research in this area is focused around using two cameras, which mimics the human and countless other life forms' visual system. This approach to computer vision is known as Stereo Vision. This chapter covers some basic knowledge about this subject. The content of this chapter would be simple enough so that the reader can understand the basic concepts required in this thesis.

There are a variety of challenges in stereo vision from camera calibration before image acquisition to the extraction of useful information from acquired images. However, from a computational complexity point of view, the most challenging issue is the Stereo Matching Problem. Stereo matching is the core research problem in this thesis. Here we cover the related work and the previous research surrounding this subject.

Computer Vision research aims to process the acquired images to create a representation of the environment and the objects residing in that environment [81]. This can be achieved with a single view. There are industrial systems that use a single view's information to achieve tasks. However, by using multiple views, these applications experienced improvements in efficiency. This improvement is a direct effect of three-dimensional reconstruction of the environment, which is not possible in a single camera case (no depth perception). Most of the research in this area is around two-camera models.

Two camera models are inspired by the biological model for stereo vision [10]. The distance between the left and the right eye creates a disparity in corresponding views of the same scene, which can be translated into depth information. This displacement in the views of each eye is inversely correlated with the distance of that object from the eyes (Figure 2.1).

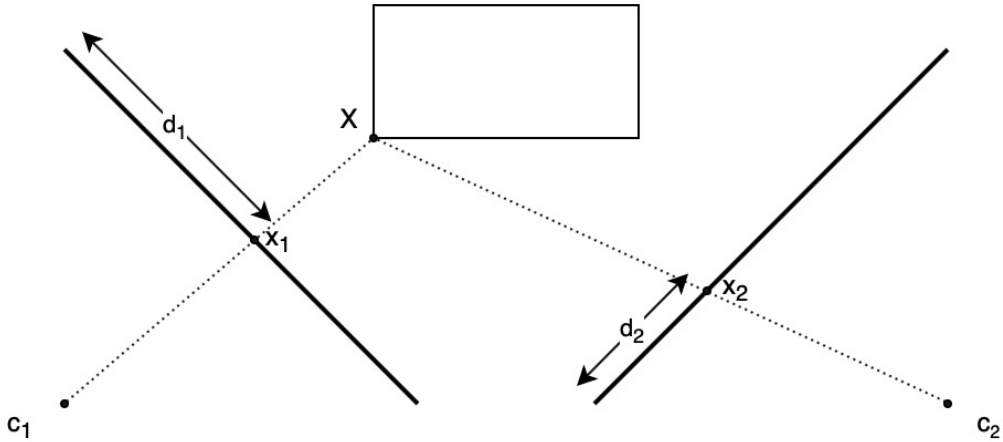


Figure 2.1: Different Cameras, c_1 and c_2 , with their representations of the point X as x_1 and x_2 respectively. The displacement of the representations can be translated to distance.

2.1 Steps in the stereo vision process

Any stereo vision processing algorithms include a common set of steps. Depending on the algorithm, some of the steps may not be required, but the order of the steps are the same regardless of the approach [12]. All of the required steps will be explained briefly in the following sections (as outlined in Figure 2.2).

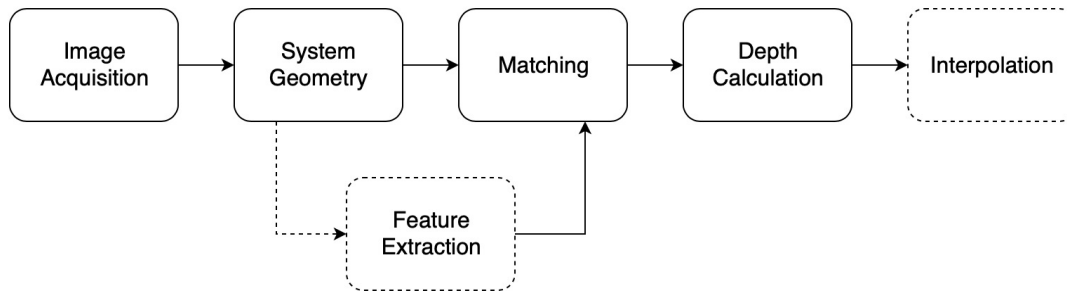


Figure 2.2: Steps Required for Stereo Vision Process. Dashed items are not required by all algorithms.

2.1.1 Image Acquisition

Several approaches can be used in image acquisition. Either two images will be captured synchronously, resulting in two representations of a scene at the same time, or two images will be captured in fixed intervals by a moving camera. Each approach is suitable for certain applications with different needs.

2.1.2 System Geometry

The system geometry is comprised of all cameras' intrinsic and extrinsic parameters alongside some between camera parameters such as baseline. The knowledge about the system geometry is important during future steps and enables us to create some restrictions for Stereo Matching problem. This knowledge is also useful for projecting the depth knowledge into the three-dimensional environment. The system geometry will be explained with more detail during future sections.

2.1.3 Feature Extraction

This step is necessary for some algorithms that rely on matching features extracted from images of the scene. The output of this step is directly used in the next step.

2.1.4 The Matching Problem

The matching problem, also known as the stereo correspondence problem, is the most important step in stereo vision. This problem consists of finding a unique map between the points or features in two different images (representations) of the same scene. The essence of this step is a search problem, and from a complexity point of view, this search problem is the most complex step in stereo vision. Additionally, the remaining steps are straight forward after a solution to the stereo correspondence problem is found. So since the output of this search process is directly affecting the final results, we will be having a deeper look at the matching problem in the remainder of this section.

2.1.5 Depth Calculation

After the matching problem is solved, to calculate the depth values we only need to do some triangulation. Also, by knowing the systems geometry and the epipolar restrictions (which will be covered in section 2.3) this calculations will be as easy as matrix multiplication.

2.1.6 Interpolation

Depending on the algorithm, if the depth map (Figure 2.3) of the previous step is sparse, interpolation is necessary to complete the depth map. However, this step is unnecessary if the result of the algorithm is a dense depth map.

2.2 Stereo Correspondence Problem

As mentioned before, the Stereo Correspondence Problem is the most important problem of the Stereo Vision. In this problem, we are trying to find that for each three-dimensional point, which point is its projection on each two-dimensional representations (images) of the scene. Figure 2.4 shows a depiction of the setup. The essence of this problem is a search problem, but there is no efficient algorithm for solving this issue, and it is still an active research subject. Deeper detail of the epipolar geometry will be explained in section 2.3.

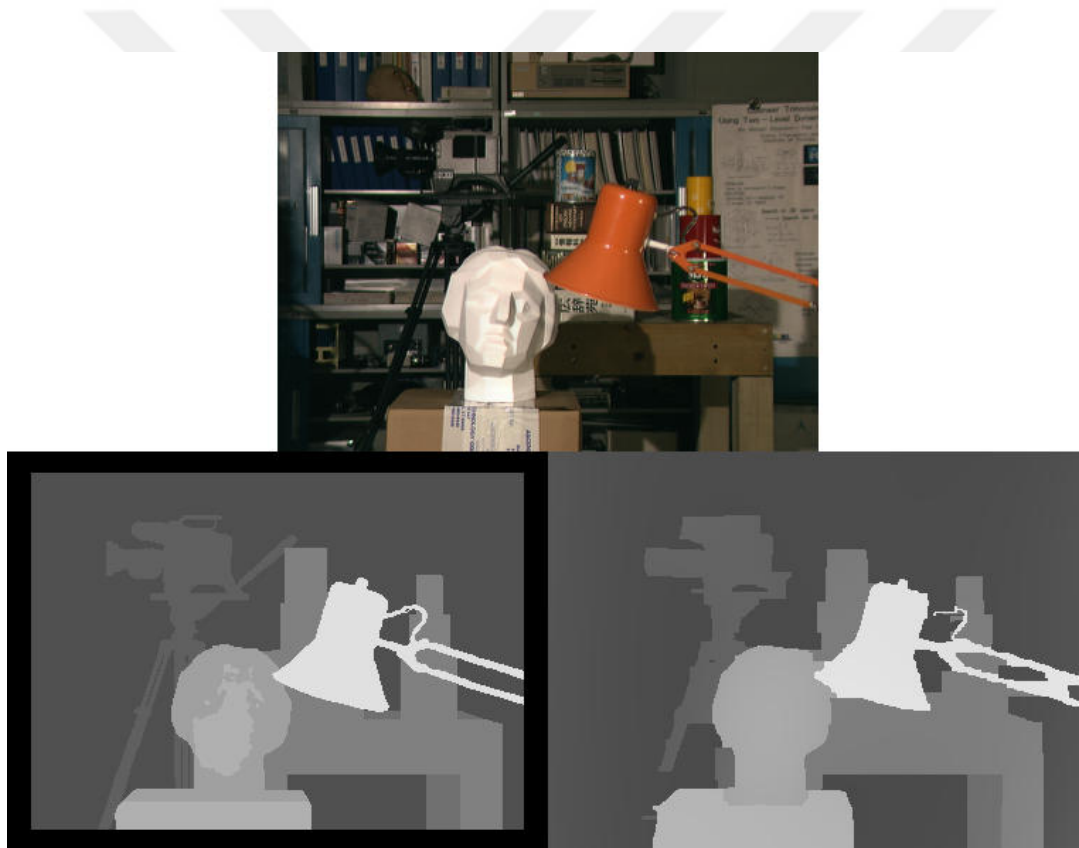


Figure 2.3: Disparity/depth map for a scene. Top: the scene. Bottom left: ground truth. Bottom right: output from an example algorithm.)

So far, the core problem of stereo vision is a searching problem for corresponding points on two images. However, a simple search of the whole image is computationally expensive, if not infeasible, and yields a very poor result. We can improve the result of the search, or the matching process, by defining some restrictions on the search parameters. A certain amount of restrictions are introduced in the literature, but the most common restrictions are mentioned below.

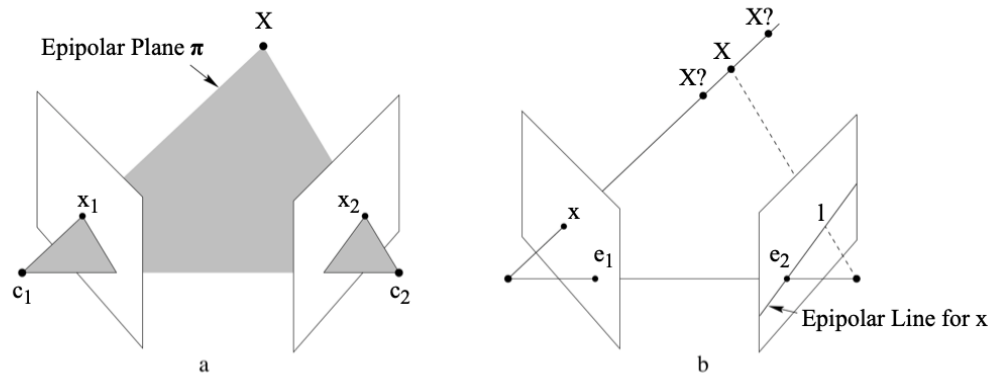


Figure 2.4: **Point correspondence geometry.** (a) c_1 and c_2 are representing the cameras. The camera centers, point X and their representations, x_1 and x_2 lie on a plane π . (b) Image point x_1 back projects into a ray in the three-dimensional space defined by the camera center and point x . The line l is the image of this ray on the other camera. Since the point X should be on this ray, its image point should be on the line l (source: [32]).

1. **Similarity Restriction:** This restriction entails that representation of a three-dimensional point on each image should have the same properties. These properties depend on the algorithm and can be the pixel intensities, shapes, edges, corners, etc.
2. **Uniqueness Restriction:** This restriction entails that each representation on one camera has one and only one matching representation on the other camera. However, applying this restriction may cause some problems. For example, in the case of occlusions, not all the three-dimensional points may have representations on both cameras.
3. **Disparity Continuity Restriction:** This restriction entails that the resulting disparities from the algorithm are usually smooth with as little as possible discontinuities expected.
4. **Epipolar Restriction:** This restriction helps the search process by reducing the area on the image that is expected to have the matching representation. This is a very important restriction because this restriction is only related to the system geometry and not the features of each image. This restriction will be explained in more detail section 2.3.

Depending on the algorithm, any number of these restrictions can be applied in different orders. Moreover, these restrictions may be named differently by other authors or may be combined under the same branch, but they are representing the same restrictions.

2.3 Epipolar Geometry

Up to this point, we briefly explained each step and process in stereo vision. The key assumption we made was that calculating the depth information of the three-dimensional scene is possible if we have two (or more) two-dimensional representations (images) of it. In this section we will explain the theory behind this assumption. This section is not a complete overview of the theory since it needs much more detailed explanations than we can achieve in this work. However, we will try to touch the key points of the theory and, we will introduce extra reading material for the readers who want a deeper understanding of the subject.

It is essential to have a clear definition of epipolar geometry. Hartley and Zisserman, authors of "Multiview Geometry In Computer Vision" [32], define the epipolar geometry as follow:

“The epipolar geometry between two views is essentially the geometry of the intersection of the image plains with the pencil of the plains having the baseline as axis. The baseline is the line joining the camera centers.”

The image plans, baseline, and the plain, including baseline and the three-dimensional point X , are visible in Figure 2.5.

The two views mentioned can either be acquired at the same time by using two cameras (a stereo rig) or at a fixed interval using a moving camera. Both views have a camera matrix associated with them. We name these matrices P_1 and P_2 for the first and second view, respectively. For a point, X in the three-dimensional space, its representation in each view can be calculated using its camera matrix,

$$\begin{aligned}x_1 &= P_1 X \\x_2 &= P_2 X\end{aligned}\tag{2.1}$$

x_1 and x_2 are defined as corresponding points since they are representing the same three-dimensional point, X , from different views.

2.3.1 Important Questions

In their book, Hartley and Zisserman [32] address three questions:

1. Given a point x_1 in one view, how can we constrain the location of the corresponding point x_2 in the other view?
2. Given a set of corresponding points from both views $\{x_1^i \leftrightarrow x_2^i\}, i = 1 \dots n$, can we calculate the camera matrices P_1 and P_2 ?
3. Given two corresponding points $x_1 \leftrightarrow x_2$ from two views and camera matrices P_1 and P_2 , can we calculate the location of X (the original point) in three-dimensional space?

The epipolar geometry answers the first question. A point in one view defines a line, known as an epipolar line, on the other view which the corresponding line resides. As we have mentioned before, this restriction of the location of the corresponding point is one of the most important restrictions in the stereo correspondence problem, and it reduces the search issue's complexity. The solution for the second question is useful for calibration of cameras in a stereo vision setup. However, the camera calibration is outside the bounds of this research, and throughout the research, we will assume that the camera and system geometries are known. The solution for the third question is directly related to the three-dimensional reconstruction of the scene.

2.3.2 Epipolar Restriction

Imagine X is a point in three-dimensional space and its projections in two views are named x_1 and x_2 . As depicted in Figure 2.4a, the three-dimensional point X , its representations x_1 and x_2 , and camera centers c_1 and c_2 are all on a plane called epipolar plane, denoted by π (Figure 2.5). Knowing that this plane exists is paramount to locating the correspondence point.

Now imagine that we only know the location of x_1 . We know 3 out of the 5 points mentioned in the last paragraph. These points are two camera centers c_1 , c_2 , and x_1 . Using these we can calculate the epipolar plane π . From the above, we also know that the correspondence point x_2 lies on π . Hence, x_2 lies on the intersection on second image view and π . This intersection line is called the epipolar line of x_1 , denoted by l_2 .

Going back to epipolar restriction, we only need to search alongside the epipolar line l_2 to find the corresponding point of the x_1 .

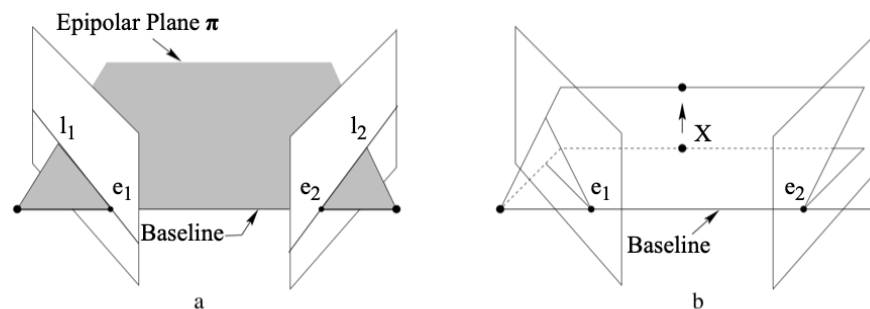


Figure 2.5: **Epipolar Geometry.** (a) Any plane π containing the baseline is an epipolar plane and its intersections with images are epipolar lines l_1 and l_2 . (b) Since the location of the point X varies, the epipolar planes rotate with the baseline as their axis. All epipolar lines intersect at the epipoles, denoted by e_1 and e_2 . (source: [32]).

Table2.1: Summary of fundamental matrix properties

Property	
Point Correspondence	$\forall x_1 \leftrightarrow x_2 \exists F : x_2^T F x_1 = 0$
Epipolar Lines	$l_2 = F x_1$ $l_1 = F^T x_2$
Epipoles	$F e_1 = 0$ $F^T e_2 = 0$

2.4 Algebraic representation of epipolar geometry

In the previous section, we derived that in a pair of images, for any point x_1 in the first view there exists an epipolar line l_2 in the second image and for any point, x_2 in the second view there exists an epipolar line l_1 in the first image. This shows that there is a map

$$x_1 \mapsto l_2 \quad (2.2)$$

from a point in one image to a line in the other image. In this section, we will explore the nature of this mapping.

2.4.1 The fundamental matrix F

Let us imagine we have two images that are acquired by two-cameras with different centers; then the fundamental matrix F is a 3×3 homogeneous matrix of rank 2 which satisfies

$$x_2^T F x_1 = 0 \quad (2.3)$$

for all corresponding points $x_1 \leftrightarrow x_2$.

We briefly review the properties of the fundamental matrix in table 2.1. The reader can refer to [32] for a derivation of proofs. The Fundamental matrix can be used in image rectification process, and it may be used to retrieve the camera matrices. Hence it is a crucial part of any stereo system.

2.4.2 Estimation of fundamental matrix F

To enforce the epipolar restriction mentioned in section 2.3.2, a good estimation of the fundamental matrix is needed. The fundamental matrix can be calculated from a pair of images obtained from a very specific camera motion (refer to [32] for more detail). However, it is also possible to estimate the matrix from a set of corresponding points from the views.

From the definition of fundamental matrix we know that for any pairs of corresponding points $x_2 \leftrightarrow x_1$, Equation 2.3 is satisfied. By writing x_1 and x_2 in homogeneous

coordinates

$$\begin{aligned}\mathbf{x}_1^T &= (x_1, y_1, 1) \\ \mathbf{x}_2^T &= (x_2, y_2, 1)\end{aligned}\tag{2.4}$$

and simplifying the equation above we will have

$$x_1x_2f_{11} + y_1x_2f_{12} + x_2f_{13} + x_1y_2f_{21} + y_1y_2f_{22} + y_2f_{23} + x_1f_{31} + y_1f_{32} + f_{33} = 0\tag{2.5}$$

To simplify, we write the Equation 2.5 as the following inner product

$$(x_1x_2, y_1x_2, x_2, x_1y_2, y_1y_2, y_2, x_1, y_1, 1) f = 0\tag{2.6}$$

where

$$f = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})\tag{2.7}$$

From a set of n matching points, we can obtain a set of linear equations

$$Af = \begin{bmatrix} x_1^{(1)}x_2^{(1)} & y_1^{(1)}x_2^{(1)} & x_2^{(1)} & x_1^{(1)}y_2^{(1)} & y_1^{(1)}y_2^{(1)} & y_2^{(1)} & x_1^{(1)} & y_1^{(1)} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{(n)}x_2^{(n)} & y_1^{(n)}x_2^{(n)} & x_2^{(n)} & x_1^{(n)}y_2^{(n)} & y_1^{(n)}y_2^{(n)} & y_2^{(n)} & x_1^{(n)} & y_1^{(n)} & 1 \end{bmatrix} f = 0\tag{2.8}$$

If matrix A has a rank of lower than or equal to eight, then a solution exists, and if the rank is exactly eight then the solution is unique. Some algorithms can be devised to automate this process by only taking two images as input without any extra information. Some of the common steps [32] involved in this automated process are:

1. Isolating matching points on both images. This can be achieved by feature matching algorithms such as [52] and [7].
2. Using the matching points to find an initial solution to equations in matrix A .
3. Utilizing an iterative approach with the initial solution as a seed to find a more accurate solution.

Since the fore-mentioned topic is out of this research's bounds, the reader can refer to [32] for a detail explanation of possible approaches to estimating the fundamental matrix F .

2.5 Image Rectification

In the section 2.4.1, we quickly mentioned that the fundamental matrix F can be used for image rectification process. The goal of this process is to do an affine transformation on both images in a way that their epipolar lines become parallel with the x -direction. These horizontally aligned epipolar lines are also called scan-lines because each row in both images can be scanned for matching points.

This affine transformation is equivalent of rotating, and/or translating the cameras, so they are completely parallel with each other. Both of the image planes will be lying on a plane named rectified plane. The resulting images have a unique and advantageous property. The epipolar lines are alongside the x -direction and the corresponding points in two images will be alongside the same horizontal line. Figure 2.6 illustrates this projection's transformation. In this scenario, the search space for each corresponding point becomes very limited, which reduces the complexity of the matching process.

Since there can be an infinite number of rectified planes, there is no unique algorithmic solution for the rectification process. [33], [50], [25], and [56] are some of the proposed algorithms for rectifying a pair of images. However, because the images resulting in the rectification process are deformed and warped, and this causes the images to lose detail, the focus is to achieve the rectification with the least amount of deforming.

2.6 Matching Algorithms

From the previous discussion about the stereo matching problem in the section 2.2, we know that some restrictions should be imposed on the system's geometry to find a solution for the matching problem, and these solutions may require high computational capacity. A common approach to Matching algorithms is to correspond the pixels of one image to the pixels of the other image [15].

Matching algorithms can be divided into different categories depending on their attributes. A high level division [15, 12] could be as follows:

1. **Local Methods:** In these types of methods, the restrictions will be applied to a limited number of pixels in the target pixel's vicinity. These methods are not considered expensive from a computational point of view. However, they result in a less coherent depth map since their knowledge of the image is very limited at each step.
2. **Global Methods:** In these methods, the restrictions are either applied to the entire image or applied to a meaningful portion of the image such as a scan-line. They yield much better results in comparison to local methods, especially in some regions that have less detail. The global methods are very expensive, computationally.

The above definitions may seem a bit arbitrary because all the existing algorithms occupy a location in between the local-global spectrum. So a binary classification of them requires some amount of arbitrary definitions. In the following section, We will briefly describe some of the popular methods of stereo matching in the literature.

Table 2.2: Stereo Matching Methods

Method	Type	References
Block Matching	Local	[3]
Block Matching with Adaptive Window	Local	[40]
Feature Matching	Local	[94]
Dynamic Programming	Global	[8, 19, 67]
Intrinsic Curves	Global	[92, 93]
Graph Cuts	Global	[13, 45]
Belief Propagation	Global	[91]

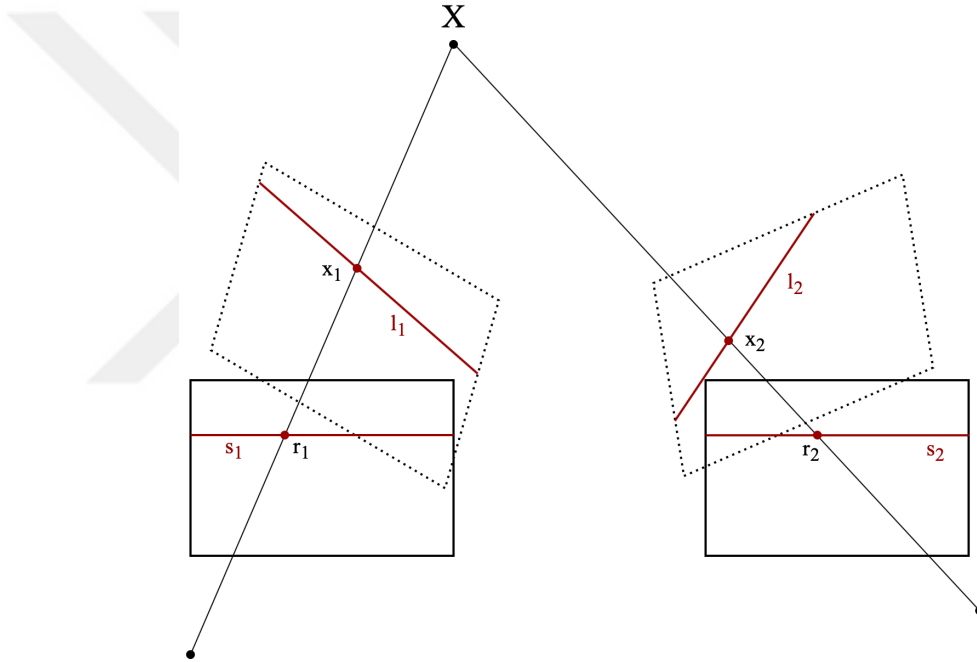


Figure 2.6: Image rectification setup. x_1 and x_2 are images of X . r_1 and r_2 are the locations of x_1 and x_2 on the rectified images. Epipolar lines, l_1 and l_2 , are converted to horizontal lines, s_1 and s_2 , in the rectified images.

2.6.1 Block Matching

The block matching algorithm tries to estimate the disparity map by comparing a small patch of the first view with a series of patches with the same size from the second view. This approach is classified as a local method. In block matching, the epipolar restriction is very important, and it reduces the search space to a one-dimensional search (Figure 2.7).

Different metrics are used by the algorithm for matching the patches. The most popular ones are Zero Normalized Cross-Correlation (ZNCC), Normalized Cross-Correlation (NCC), Sum of Squared Differences (SSD), Normalized Sum of Squared

Differences (NSSD), Sum of Absolute Differences (SAD). Normalized Cross-Correlation

$$\frac{\sum_{u,v}(I_1(u, v) - \bar{I}_1).(I_2(u + d, v) - \bar{I}_2)}{\sqrt{\sum_{u,v}(I_1(u, v) - \bar{I}_1)^2.(I_2(u + d, v) - \bar{I}_2)^2}} \quad (2.9)$$

is the most used in the literature. It is normalized and more sensitive to the number of changes in intensities. On the other hand, the Sum of Squared Differences (SSD) and Sum of Absolute Distance (SAD) are efficient and simpler computationally. Aschwanden and Guggenbuhl [3] have robustly compared all these metrics together.

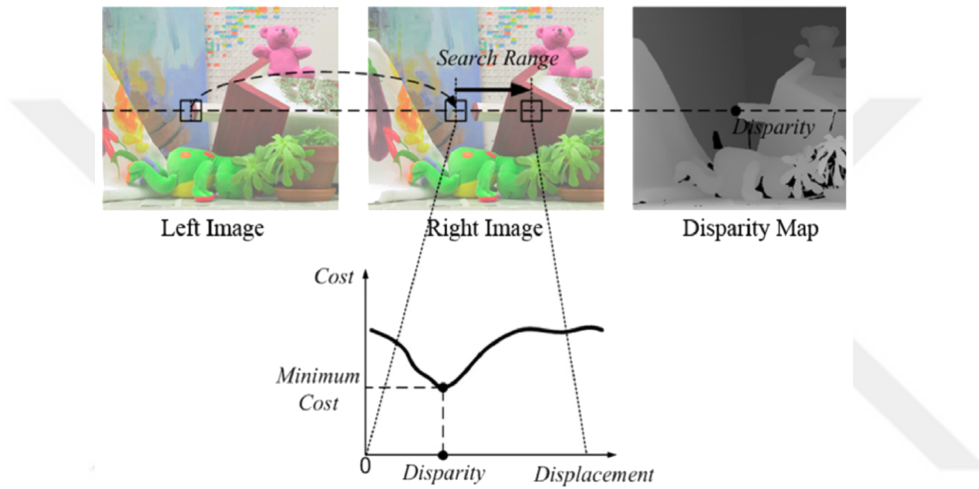


Figure 2.7: Block matching on two rectified images (source: [86]).

2.6.2 Block Matching with Adaptive Window

The size of the small patch around the target pixel in block matching is an important factor in avoiding mismatches due to featureless areas or patches of similar patterns. This window should be large enough to have adequate variation in disparities for a good match, yet small enough so it will not be affected by projective distortions [40].

The window size can be optimized for the best size given the amount of variation is in the window. This can be achieved by iteratively expanding the size of the window when its variation is less than a threshold.

2.6.3 Feature Matching

The block matching approach is sensitive to depth discontinuities since the region around them has pixels from more than one depth value. As mentioned before, It is also sensitive to featureless regions with homogeneous texture. The feature matching methods work with different low-level image features such as edges, curves, and corners. A big issue related to feature matching approaches is that they produce a sparse depth map and due to the need for dense maps in many applications, and improvements in efficient block matching algorithms, the interest for these methods are

declined. Two types of popular feature-based stereo matching algorithms will be explained here: hierarchical feature matching and segmentation matching.

Hierarchical feature matching algorithms utilize different types of features such as lines, vertices, edges, and surfaces. Venkateswar and Chellappa [94] have proposed an algorithm that starts from matching the highest level of features and moves to lower features. The algorithm initially extracts the features and creates a graph containing them then match the features from top to bottom.

Another approach is to segment the image into different sections and then try to match these segmented parts with each other. This approach, unlike the previous approach, yields a dense depth map. However, this method is sensitive to the quality of the segmentation algorithm.

2.6.4 Dynamic Programming

Dynamic programming is a computer science approach to solving optimization problems by dividing them into smaller sub-problems [18]. Using epipolar constraint, we can determine the cost function as a minimum cost path through the disparity space image (DSI). Figure 2.9 shows a depiction of DSI. The Optimal path can be obtained recursively using partial paths.

Ohta and Kanade [67] and Cox et al. [19] defined the DSI by using each view's scan-lines as spacial axis, then using the dynamic programming to find the best path from bottom left corner or top-right corner.

In addition to optimizing the cost function for each scan-line, some constraints between the neighboring scan-lines can be used to reduce the ambiguity. Ohta and Kanade [67] try optimizing a two-dimensional area around the scan-line. They have integrated the between scan-line optimization into the original optimization process. Belhumeur [8] approached this issue in two stages. First, optimizing the cost function for each scan-line then smoothing disparities between the scan-lines. Cox et al. [19] proposed to reduce the inconsistencies between scan-lines by penalizing the discontinuities.

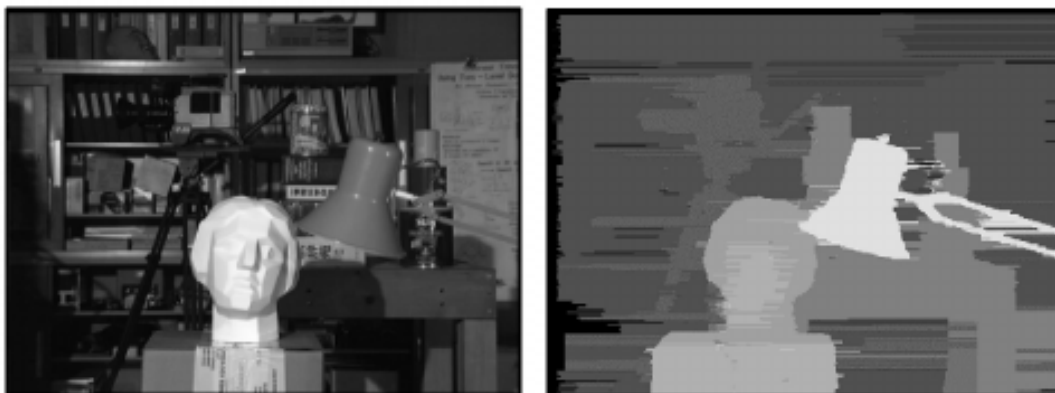


Figure 2.8: Disparity map (right) created from image (left). The horizontal artifacts are the byproduct of optimizing each scan line separately.

The primary problem of the dynamic programming approach is the possibility that the local error may propagate alongside a scan-line [15]. The horizontal artifacts created by this problem are visible in figure 2.8.

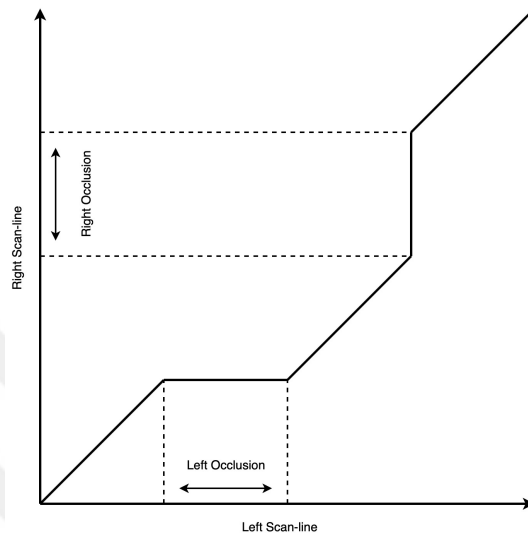


Figure 2.9: Disparity Space image. The solid line in the image represents valid matches between the left and right scan-lines. This model inherently can detect occlusions. (source: [19])

2.6.5 Intrinsic Curves

An alternative approach to finding an optimum map between two scan-lines is proposed by Tomasi and Manduchi [93] using a features vector called an intrinsic curve. An intrinsic curve is a vector representation of image descriptors. These descriptors are defined by different operators such as edge and corner operators to a scan-line.

An intrinsic curve $C \in R^n$ can be defined as

$$C = \{p(x) \mid x \in \mathbb{R}\} \quad (2.10)$$

$$p(x) = (p_1(x), p_2(x), \dots, p_n(x))$$

Since the intrinsic curve consists of vectors, which in turn they consist of feature, this method is technically a feature-based method. We formulate the search for disparities in intrinsic curve space as the nearest neighbor problem. Any residual ambiguities, especially in between the scan-lines, are solved by optimizing a global metric using dynamic programming. See [92] and [93] for more detail.

2.6.6 Graph Cuts

A primary issue of using dynamic programming methods is that they cannot use strong continuity constraints on both horizontal and vertical axes. To use both these

constraints strongly, we can formulate the matching problem as a maximum flow problem on a graph [18].

We define a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The vertex set is

$$V = V^* \cup \{s, t\} \quad (2.11)$$

where s is the source and t is the sink and

$$V^* = \{(x, y, d) \mid x \in [0, x_{\max}], y \in [0, y_{\max}], d \in [0, d_{\max}]\} \quad (2.12)$$

and the edges are defined as

$$E = \left\{ \begin{array}{l|l} (u, v) \in V^* \times V^* & \|u - v\| = 1 \\ (s, (x, y, 0)) & x \in [0, x_{\max}] \\ ((x, y, d_{\max}), t) & y \in [0, y_{\max}] \end{array} \right\} \quad (2.13)$$

Each node has a cost value, and each edge has a flow capacity associated with them. The flow capacity is defined as a function of its adjacent nodes. A cut is defined as a partition of the vertex set V into two subsets, separating the source from the sink. The capacity of a cut is simply the sum of the edge capacities that make up the cut. The cut with minimum capacity / maximizes the flow through the graph (Figure 2.10). This minimum cut is analogous to the best path in the dynamic programming approach but extended into three-dimensions.

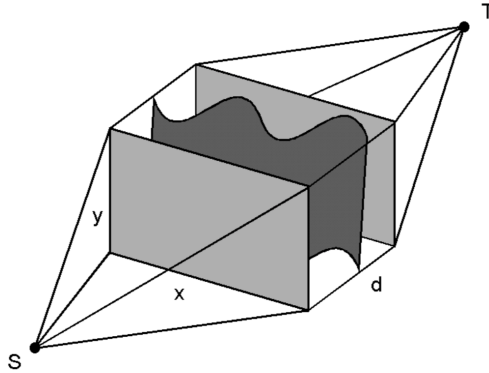


Figure 2.10: A representation of max-flow/min-cut setup in stereo matching problem. Source and sink are represented by S and T respectively (Source [15]).

Usually, the graph cut method is expected to have greater computational complexity than dynamic programming. Fortunately, many efficient solutions are created for this approach. Boykov and Kolmogorov [13] have created a Ford-Fulkerson style augmenting paths algorithm which is much faster than standard approaches. Kolmogorov and Zabih [45] proposed a novel graph architecture in which the vertices are pixel correspondences instead of pixels themselves, and they make it possible to enforce uniqueness restrictions to handle occlusions.

2.6.7 Belief Propagation

The stereo matching problem can be modeled using Markov Random Fields (MRF). MRF is a sort of graphical models using undirected graphs which can have loops in them. Figure 2.11 illustrates stereo matching problem modeled with MRF.

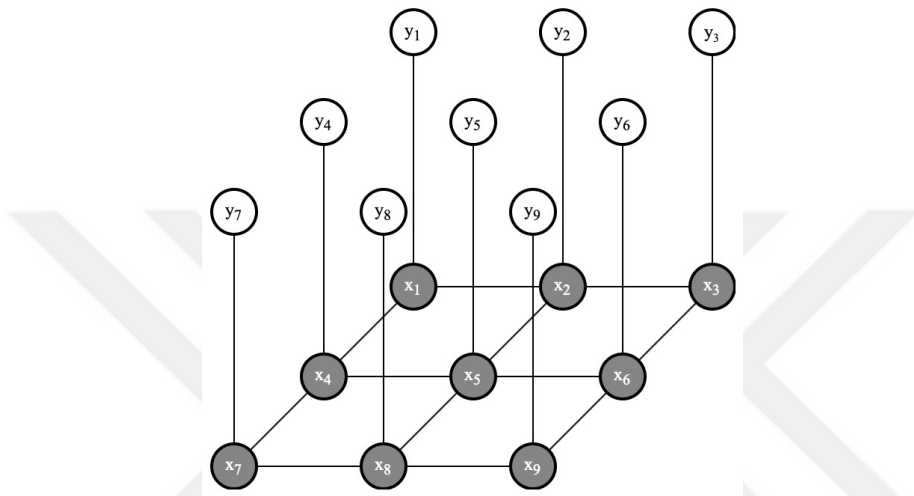


Figure 2.11: Stereo modeled with MRF. The white vertices, denoted by $Y = \{y_1, \dots, y_n\}$, are the observed variables (pixel intensity values). The highlighted vertices, denoted by $X = \{x_1, \dots, x_n\}$, are the hidden variables, which represents the disparity values. The hidden variable values are more generally referred to as labels. (Adapted from [91])

The edges of this graph represent a dependency for each vertex. For example, in the model in Figure 2.11 each hidden vertex, only depends on its four closest neighbors. This is called a Markov assumption.

The Belief Propagation (BP), like all the global methods mentioned before, optimize a cost function to solve the stereo matching problem. The cost function can be formulated as

$$C(X, Y) = \sum_i D(x_i, y_i) + \sum_{j \in N(i)} S(x_i, x_j) \quad (2.14)$$

where D is the data cost of matching pixels from two images, S is the smoothness cost of the adjacent hidden vertices that ensures the smoothness of the disparity, and $N(i)$ is the set of all the neighbors of the hidden vertex i . Observed and hidden variables are denoted by Y and X respectively.

Belief propagation is a message passing algorithm; it means that a vertex, i , passes a message to neighboring vertex, j , containing its belief about j 's label after gathering all the messages from its neighbors (Figure 2.12). A detailed explanation of the algorithm can be found at [91].

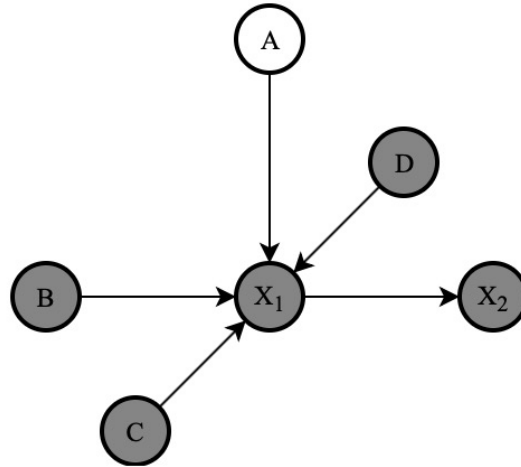


Figure 2.12: Message Passing in Belief Propagation. Vertex x_1 waits for messages from A, B, C, D then sends its message to x_2 . Note that it does not send the message from $x_2 \rightarrow x_1$ back to x_2 (Adapted from [91]).

2.7 Summary

The main goal of stereo vision research is to extract extra information from two-dimensional representations (images) of a three-dimensional scene. This additional information is the depth values of objects in the scene. By studying such systems' geometry, we know that there is an inverse correlation between the disparity of the objects in two scenes and their distances from the cameras.

There are multiple steps involved in extracting depth information. The main issue within the approach to extract information from stereo images is the matching problem. This problem, also known as the stereo correspondence problem, is computationally expensive and there is no efficient algorithm to solve it. The solution for this problem effectively determines the end approximation of the depth map.

It is shown that a naive approach to this problem is both inefficient and yields inaccurate results. Hence, the ingenuity is necessary to create algorithms that approximate a solution. Moreover, the fact that this problem is being researched for decades shows that this is not an easy problem to solve.

Many algorithms are proposed to solve the problems. Some of the noteworthy ones are Graph Cut and Belief Propagation, which try to approximate the depth map by optimizing an energy function.



CHAPTER 3

MACHINE LEARNING

In the earliest days of Artificial Intelligence (AI), the focus of researchers was on problems that were very difficult for humans to solve but were very easy for the machines to do. They involved any problem that can be formulated with a sequence of formal and mathematical rules. However, as the field grew, researchers realized that the real challenge of AI is to solve the tasks that are easy for humans to do, but they are hard to describe or defined in an axiomatic manner, such as detecting various objects in images or understanding speech.

The main task that stereo vision research is trying to solve, fits this description perfectly. Understanding depth from binocular (stereo) vision is something that comes very easy for a human but, as discussed before, has been very hard to describe and solve by the researcher formally. This problem is intensified when we realize that humans combine their spatial awareness of the object in their view (even in case of occlusions) with the information they receive from their eyes to understand the depth and create a representation of the world in their mind. These ideas will be discussed more deeply in future sections.

Engineering a solution for these challenges using logic and formal languages have been proved very difficult ¹. Researchers struggled to hard-code world's properties, even the relatively simple ones, with a set of logical statements and formal languages. They realized that the solution to this problem is to let the machine extract and accumulate the knowledge it needs by itself. The ability of the machine in creating its knowledge-base by extract these patterns from raw data is called Machine Learning (ML). This solution eliminated the need for humans engineering and hard-coding every detail into the system.

Some Machine Learning systems are designed in a way so they can learn the world and its features in a hierarchical concept. In this way, the higher-level concepts will be built on lower-level concepts. We can create a graph which maps these concepts' relations relative to each other. Such a graph would contain many layers and it would be deep. Hence this approach to AI is called Deep Learning (DL) [29].

Deep Learning is classified as a specific Machine Learning (ML) approach. The Vaan diagram in Figure 3.1 depicts this classification. It is not possible to discuss Deep Learning without having knowledge of Machine Learning in general. The readers who have an understanding of Machine Learning can skip this chapter. This chapter

¹ A famous example is the Cyc project [48].

includes a general view of Machine Learning and is not meant to have deep discussions. We will point the reader to other sources for a complete explanation if and when required.

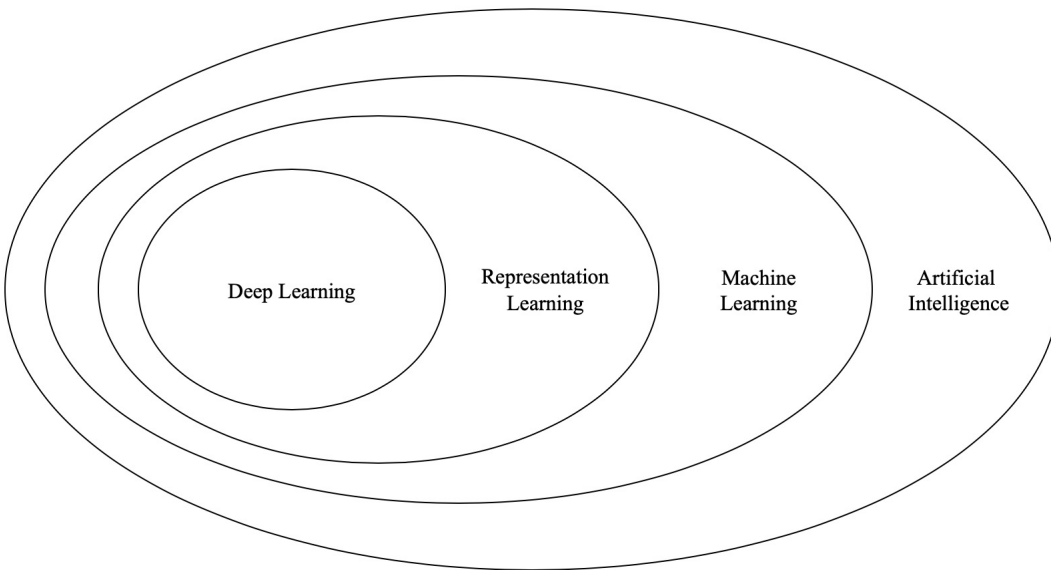


Figure 3.1: Relation of Deep Learning to Machine Learning and AI. Adapted from [29]

3.1 Representation Learning

A simple ML algorithm can learn to predict the occupancy of a building unit [9], predict people with a chance of colon cancer [37], and many other applications..

These simple algorithms can achieve tasks by receiving an interpretation, or representation, of the data. These representations are pieces of relevant information made available to the algorithm. For example, simple Machine Learning algorithms can learn to predict the occupancy of a building unit using representations such as temperature, CO2 levels, and humidity [71]. These data are presented to the algorithm, and the algorithm does not observe and examine the room directly for occupancy. This approach, as helpful as it is, does not have any control over how the representation or how they are extracted.

Choosing the right representation is very important in the success of a machine learning algorithm. For example, arithmetic operations are easier for human when the numerical system is in decimal format rather than any other formats such as hexadecimal or binary. Many AI tasks can be solved by simple algorithms given that the data representation is fitting that specific task. However, knowing which representations are useful is not always easy.

A solution to this problem is to task the machine itself to learn useful representation from the raw data. This solution is called Representation Learning. This section of the Machine Learning is complex by itself and, we highly recommend reading the

review of Representation Learning by Bengio et al. [9] for a more comprehensive discussion.

A major problem with tasking the computer to extract the information from raw data is that in reality a set of data is affected by many factors of variation and tasking an algorithm to extract high-level representations of it is not always feasible. For example, a person's silhouette depends on the angle of view, a person's pose, clothing, etc. A solution to this problem is Deep Learning.

3.2 Deep Learning

Deep Learning is an approach to solve this fundamental problem of extracting meaningful representation from raw data by defining each high-level representation in terms of lower-level representation (Figure 3.2). This structure can be as deep as necessary.

Arguably, the most famous example of Deep Learning is feed-forward Deep Neural Networks (DNN), or Multilayer Perceptron (MLP). Perceptron is a mathematical function mapping a set of inputs into a single output.

There is a deeper discussion about Deep Neural Networks, their theories, applications, and types in the methodology chapter. However, to understand the deep learning well, an understanding of Machine Learning Basics is required. We assume that the reader is familiar with some concepts of applied mathematics, such as Linear Algebra, Probability, and Calculus. Additionally, some knowledge of Numerical Computations will be helpful but not necessary.

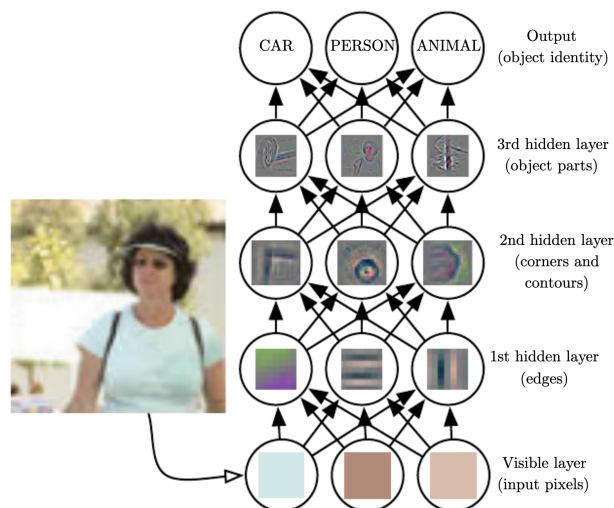


Figure 3.2: Building representations in a hierarchy by deep neural networks. Each representation will be build from lower representation, extracted from raw data (images). (source: [29])

3.3 Defining a Learning Algorithm

In his book titled "Machine Learning" [59], Mitchell has defined a learning algorithm as:

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .”

This is a very general definition. Some explanation is required to make this definition more understandable for readers who do not have a prior familiarity with Machine Learning Concepts.

3.3.1 The Task T

As mentioned before, the goal of machine learning is to attempt to solve tasks that are easy for humans to do, but difficult to define for a machine using some formal language and logical structure. Interestingly, Task T is not the process of learning. For example, when an algorithm learns to predict breast cancer, the task is prediction breast cancer, not learning to predict breast cancer. Learning is just a means of achieving it.

Goodfellow et al. [29] have composed an example set of tasks that can be achieved by machine learning methods:

1. **Classification:** These types of tasks require the algorithm to map each input into one of the k predefined classes

$$\begin{aligned} y &= f(x) \\ f : \mathbb{R}^n &\rightarrow \{1, \dots, k\} \end{aligned} \quad (3.1)$$

where x is a n dimensional feature vector and y is a category identified by a number. the output y can also be a vector of probability values which shows the probability of x belonging to each class.

2. **Classification with missing input:** Sometimes, due to the nature of some problems, it is not guaranteed that all input vectors have all the values available. In this case, instead of learning one function f , the algorithm must learn a set of functions

$$f = (f_1, \dots, f_m)^T \quad (3.2)$$

where each function f_i should learn to accept a subset of the features as an input.

3. **Regression:** These type of tasks require the algorithm to map each input into a single numerical value

$$\begin{aligned} y &= f(x) \\ f : \mathbb{R}^n &\rightarrow \mathbb{R} \end{aligned} \quad (3.3)$$

4. **Transcription:** These type of tasks require the algorithm to take a relatively unstructured data and return a discrete textual format. An example of these algorithms is Optical Character Reconstruction (OCR), which converts a photograph containing text into a string of characters.
5. **Machine Translation:** In this kind of tasks, the algorithm is asked to convert a sequence of characters into another sequence of characters. This is commonly used in natural language processing.
6. **Structured Output:** These types of tasks requires the algorithm to map each input into a multi-part output (like a vector) that contains a relationship between its different elements. An example of these algorithms is the picture captioning programs that take the picture as input and outputs a coherent sequence of words that makes a meaningful sentence.
7. **Anomaly Detection:** These type of tasks require the algorithm to scan a set of inputs and decides which ones are anomalies.
8. **Synthesis and Sampling:** These type of tasks ask the algorithm to learn a set of input values and then be able to synthesize a previously unseen example.
9. **Imputation of missing values:** These types of tasks require the algorithm to predict a missing value x_i given an input $x \in \mathbb{R}^n$.
10. **Denosing:** These types of tasks require the algorithm to map a faulty input $\hat{x} \in \mathbb{R}^n$ into a pristine input $x \in \mathbb{R}^n$

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^n \quad (3.4)$$

11. **Probability Mass Function Estimation:** These types of tasks require the algorithm to estimate a Probability Mass Function (PMF)

$$p_{\text{model}} : \mathbb{R}^n \rightarrow \mathbb{R} \quad (3.5)$$

3.3.2 The Performance Measure P

In order to evaluate the learning process, any Machine Learning method needs a quantitative performance measure. Usually, each of the tasks mentioned above has specific measures tailored to their data structure. For most of the classification methods, one can use the accuracy of the method.

$$\text{Accuracy} = \frac{\text{Correctly Labeled Data}}{\text{All Data}} \quad (3.6)$$

A general performance measure can be the error rate of any algorithms. The error rate for classification methods is

$$\text{Error Rate} = \frac{\text{Wrongly Labeled Data}}{\text{All Data}} \quad (3.7)$$

but for regression methods (with continues data) one of many formulation of error rate, such Sum of Squared Errors

$$e = \sum_i (x_i - \hat{x}_i)^2 \quad (3.8)$$

can be used where x is the true value and \hat{x} is the predicted value.

Confusion Matrix

To come up with better measures, we can use a specific matrix, known as The Confusion Matrix. Figure 3.3 shows the matrix for binary classification. This idea can be expanded to multi-class cases as well. The confusion matrix is often used to describe the performance of classification models, and it is a good example of alternative performance measures.

Using the binary matrix (figure 3.3), we can devise four useful performance measures:

1. **Accuracy:** A measure showing how often the classifier is correct.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} \quad (3.9)$$

2. **Precision:** A measure which shows how correct is the classifier out of all the classes.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (3.10)$$

3. **Recall:** A measure showing how correct is the classifier out of all the positive classes.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3.11)$$

4. **F-Score:** A measure aimed for enabling comparing models by punishing models with extremely low Precision and High recall.

$$\text{F-Score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (3.12)$$

	Predicted: Yes	Predicted: No
Actual: Yes	True Positive (TP)	False Positive (FP)
Actual: No	False Negative (FN)	True Negative (TN)

Figure 3.3: Confusion matrix in binary classification.

Although Accuracy and Error Rate is good enough measures, there are specific cases in which they are misleading. For example, imagine the case of a classifier that detects a rare condition that is present in 0.2 percent of the population. The dataset in this scenario is called an unbalanced dataset.

Let us imagine we have a classification algorithm that correctly identifies every individual with this condition, but it also has a False Positive rate of 0.2 percent. This means the algorithm will miss identifying two people without this condition as positives. In this scenario, the accuracy is 99.8 percent. However, the precision is only 50 percent.

3.3.3 The Experience E

We can categorize the machine learning algorithm by what kind of Experience they will have during learning [29]. A broad categorization is to divide them into Supervised and Unsupervised methods. It is important to know that these terms do not have a clear definition, and their separating line is blurry.

- **Supervised Learning Methods:** These methods experience a dataset containing data points with the addition of their intended labels, also know as ground truth.
- **Unsupervised Learning Methods:** These methods experience a dataset containing unlabeled data point, and the goal is to learn useful information about its distribution.

Unsupervised Methods try to understand the probability density $p(x)$ of a distribution, or some interesting features of the distribution, by studying data points x . On the

other hand, Supervised Learning Algorithms try to predict the label y by observing data points x which is done by estimating $p(y|x)$

Although many Machine Learning algorithms experience a fixed dataset, either completely or in small chunks, there are some approaches that have different experiences. For example, Reinforced Learning is an approach to Machine Learning in which the method interacts with an environment, This experience would be fluid and dependent on the interactions.

3.4 Capacity of a Learning Algorithm

It is not easy to define the capacity of a learning algorithm since the learning algorithm itself does not have a solid and formal definition. A crude definition of a Learning Algorithm's capacity can be its ability to learn a wide variety of functions. We use the capacity to understand two important challenges in Machine Learning: Overfitting and Underfitting. Knowing what is expected from a learning algorithm is also important. The expectation from such methods is their ability to perform well on previously unseen data points, also called Generalization.

3.4.1 Training and Testing Errors

We can define and calculate an error measure on the dataset containing training data and aim to improve our learning algorithm to reduce this error measures, also known as Training Error. This is simply an optimization problem. The separating factor between Machine Learning and the Optimization problem is that in machine learning approaches, the Testing Error should be low as well. Testing error is the expected error of a previously unseen and new data. The testing data is used to evaluate the generalization capabilities of the learning algorithm. Two items are important in the process of learning (generalization):

1. Reduction of the training error
2. Reduction of the gap between training error and testing error

3.4.2 Overfitting and Underfitting

Overfitting and Underfitting are terms describing the status of a method's training.

- **Underfitting:** This happens when the learning method fails to learn the required task. Here, the model is unable to reduce the training error (Figures 3.4 and 3.5).
- **Overfitting:** This happens when the capacity of the model is so much higher than required for the data, and the model starts to memorize the data instead

of generalizing it. Here, the model is unable to reduce the gap between the training and testing error (Figures 3.4 and 3.5).

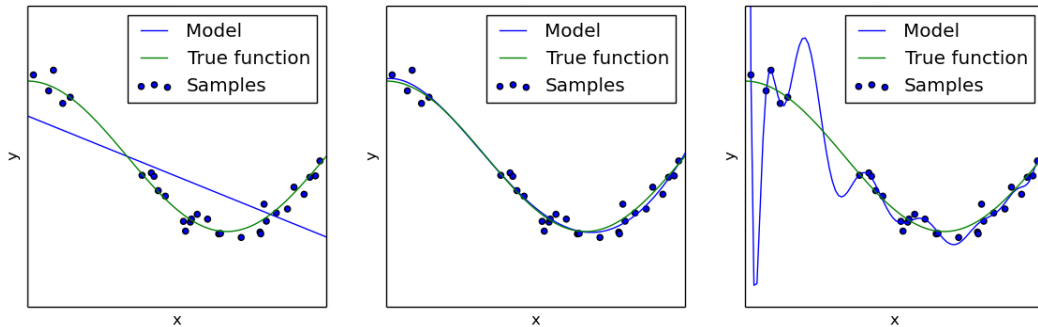


Figure 3.4: Examples of underfitting and overfitting. The data points are sampled from a function with some gaussian noise added. From left to right: Underfitting, Adequate Fitting, Overfitting.

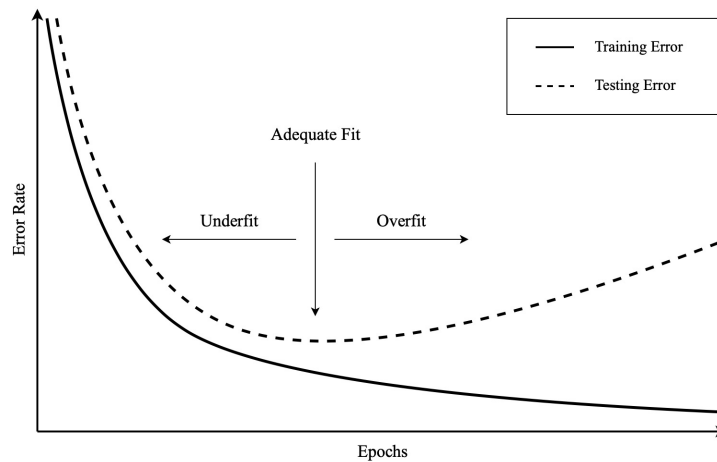


Figure 3.5: Training Error and Testing Error in different fitting scenarios.

3.5 Supervised Learning Algorithms

In section 3.3.3, we briefly mentioned that A Supervised Learning Algorithm is an algorithm that experiences a dataset containing data points x , with predefined labels as ground truth \hat{y} . The output of the algorithm will be "supervised" so its output y gets as close to \hat{y} without sacrificing generalization. In this section, some of these algorithm will be briefly introduced, but there is not enough room for an extensive explanation of each algorithm. Sources will be introduced for anyone interested in a deeper study of them.

1. **Linear Regression:** In linear regression we are trying to devise a model to predict one continuous variable Y , using other continuous variable X and a linear

model to predict \hat{Y} formulated as

$$\hat{Y} = bX + a \quad (3.13)$$

Here the task is to predict the values of $b \in \mathbb{R}$ and $a \in \mathbb{R}$. We can achieve this task by using an iterative method and minimize a cost function

$$J(b, a) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \quad (3.14)$$

$$\operatorname{argmin}_{b,a} J(b, a)$$

where $y^{(i)} \in Y$. This method is called Least Mean Squared (LMS).

The LMS tries to minimize the error between the real value Y and the predicted value \hat{Y} by changing variables a and b (Figure 3.6).

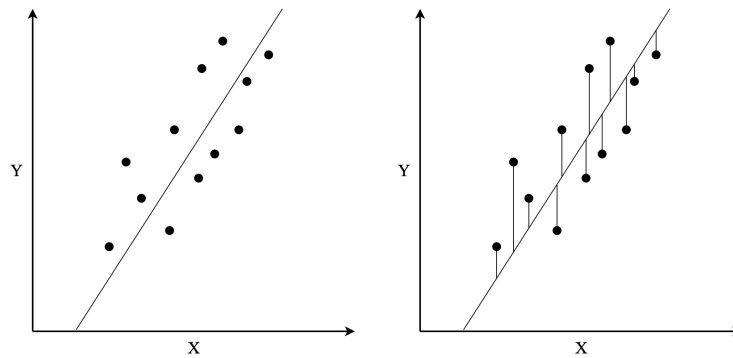


Figure 3.6: The regression line calculated by Least Mean Squared algorithm. Left: Regression line for continuous variables x and y . Right: depicting errors $\hat{y} - y$ that the algorithm tries to minimize.

2. **Logistic Regression:** In logistic regression, we are trying to devise a model to predict a variable Y that follows a binomial distribution, using other continuous variable X . It helps in finding the probability that a new instance belongs to a certain class. The output of this method is a probability value between 0 and 1. We can interpret the logistic regression as a special type of linear regression.

By applying the sigmoid function

$$p = \frac{1}{1 + e^{-y}} \quad (3.15)$$

to the linear regression we will have

$$p = \frac{1}{1 + e^{-bX-a}} \quad (3.16)$$

Figure 3.7 shows how modifying linear regression can be used for classification purposes. For additional study in the subject, refer to [70].

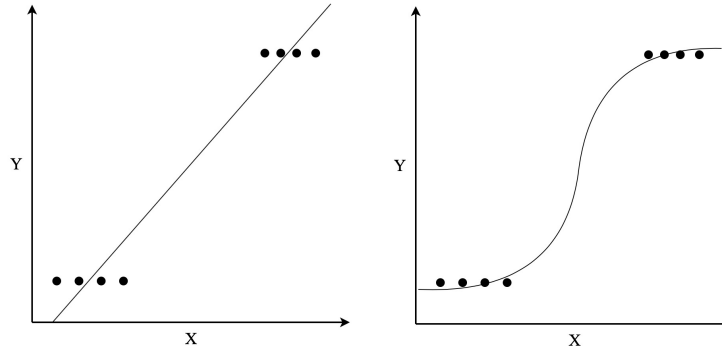


Figure 3.7: Comparing Linear Regression and Logistic Regression's performance with categorical data. According to Y , the data in the picture are in two categories. Left: Linear Regression. Right: Logistic Regression.

3. **Decision Trees:** This is a graphical method that can classify data based on creating a tree structure called a decision tree. This tree partitions the dataset into distinctive partitions, each falling under a category. Figure 3.8 is an example of a decision tree. In this approach, the learning process is to figure out which properties are the most descriptive and minimizing the generalization error.

Many trees can be induced from the same data depending on the approach and criteria. There are many algorithms inducing decision trees such as ID3 [74], C4.5 [75], CART [14].

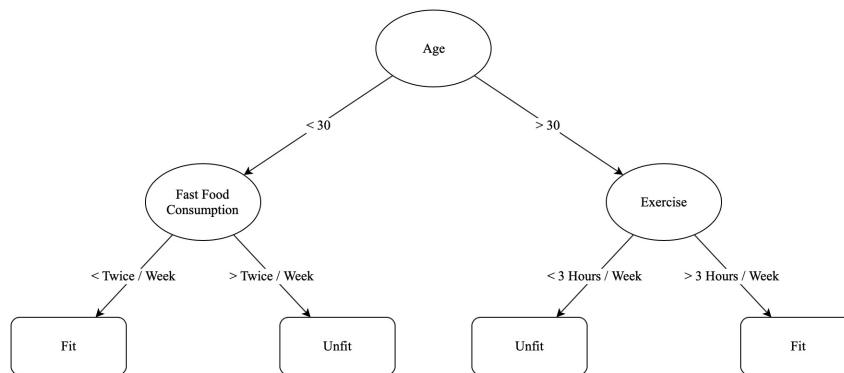


Figure 3.8: An example of a decision tree. This decision tree classifies people to two classes, fit and unfit, using Age, Fast Food consumption and Exercise.

4. **Random Forest:** The aim of the random forest approach is to significantly increase the accuracy of a classifying problem by using an ensemble of decision trees (Figure 3.9). After generating a random amount of decision trees, each tree will vote for a class, and the most popular choice will be the resulting class.

For an in-depth study of Random Forests, refer to the Ph.D. Dissertation by Loupe [51]

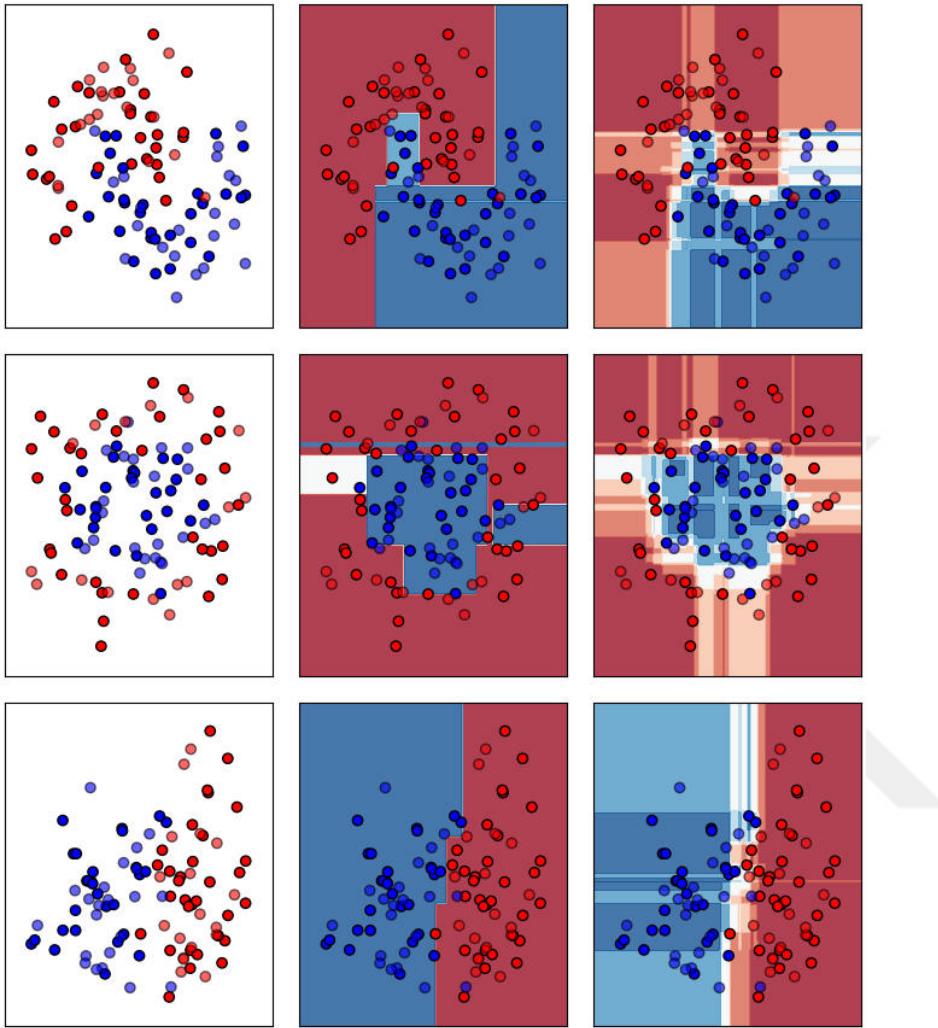


Figure 3.9: Visual Comparison of Decision Tree and Random Forest Results. Column descriptions from left to right: Generated Data, Decision Tree Results, Random Forest Results.

5. **Naive Bayes:** The Naive Bayes classifier is using the Bayes theorem to predict membership probability for each class. In this case, Naive Bayes can be thought of a Maximum A Posteriori (MAP) classifier. The promise is to maximize the posterior probability of the hypothesis

$$\begin{aligned} & \operatorname{argmax} P(H|E) \\ & \operatorname{argmax} \frac{P(H) P(E|H)}{P(E)} \quad (3.17) \\ & \operatorname{argmax} P(H) P(E|H) \end{aligned}$$

where H is the hypothesis and E is the evidence. If we replace the evidence E with data features $x_i, i = 1, \dots, n$ we will have

$$\operatorname{argmax} P(H) P(x_1, x_2, \dots, x_n|H) \quad (3.18)$$

For a more comprehensive study of this approach, check the article by Rish [77].

6. **Support Vector Machines (SVM):** Many believe SVMs are among the best of supervised learning algorithms. SVM classifies the data points by finding a hyperplane that maximizes the margin between the two classes. The vectors (data points) that are crucial for defining the hyperplane are the support vectors. What SVM tries to do is to maximize the separation margin between classes (Figure 3.10).

In the simplest mode, SVM uses a n -dimensional hyperplane to separate the data points. However, by using a kernel function, the SVM can map some data, that is not separable linearly, into a higher dimension where they can be separated by a hyperplane that translates to a non-linear region in the initial space. This is called a kernel trick. The Support Vector Machine is an exceedingly wide study subject. We recommend "Learning with kernels: by Scholkopf and Smola [85] for extra reading.

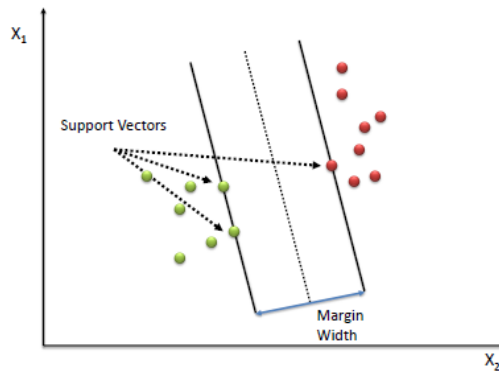


Figure 3.10: Visualizing the support vectors and separation plane created by SVM. The hyper plane of this example is one-dimensional (line) since the space is two-dimensional.

3.6 Unsupervised Learning Algorithms

A brief introduction to Unsupervised Learning Algorithms was attempted in section 3.3.3. An Unsupervised Learning Algorithm is a method that experiences a dataset containing data points x to learn either the Probability Mass Function (PMF) of the data's distribution or to find useful information about it. In this section, some of these algorithms will be briefly explained. Since there is not enough space for a comprehensive explanation, we will direct the reader to sources for more studies.

1. **Gaussian Mixture Modelling (GMM):** In this approach to unsupervised learning, the goal is to define the dataset using k Gaussian distributions (Figure 3.11). The probability for each point is determined with a mixture of Gaussian distri-

butions

$$p(x) = \sum_{i=1}^k \pi_i N(x | \mu_i) \quad (3.19)$$

The number k will not be calculated in the algorithm and will be assigned; although, it is possible to find the best variables using a brute force method. This is a probabilistic approach so the assignment of the data will be soft. The algorithm used for fitting the models into the data is called Expectation Maximization (EM). The EM algorithm consists of two steps, an E-step or Expectation step and M-step or Maximization step.

The E-step is to compute the expectations of variables, and M-step is to compute the maximum likelihood parameters according to these variables. These two steps will be repeated until convergence.

2. **K-means:** In this method, the goal is to describe the dataset using k classes (Figure 3.12). k -means can be viewed as a specific version of GMM where we are trying to fit k Gaussian distributions without taking variance and covariance into account. Additionally, the class assignments in k -means algorithm is hard, as opposed to soft assignment in GMM.

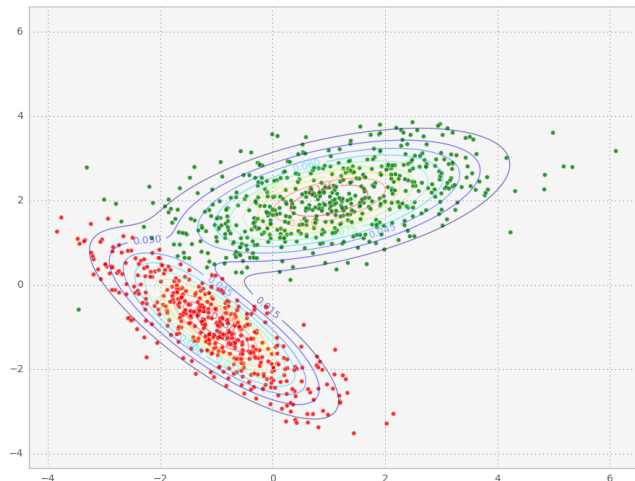


Figure 3.11: A dataset described by the mixture of two Gaussian models.

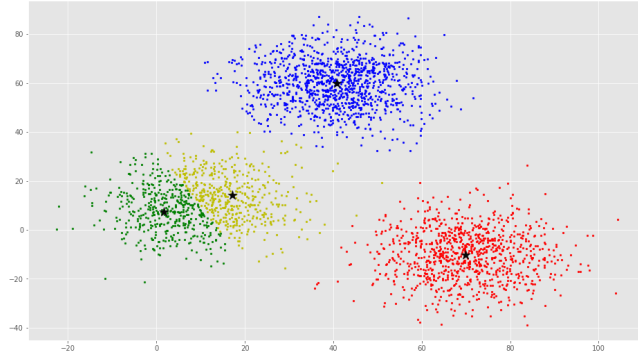


Figure 3.12: A dataset divided into 4 categories using k -means algorithm. The only important variable for each class is the mean of the class. All the means are shown with a star.

3.7 Gradient Descent

Anyone who is interested in machine learning, or deep learning, should study optimization algorithms. Optimization is a vast topic on its own, and we cannot even scratch its surface in this section. However, a quick introduction to Gradient-Based Optimization is necessary to understand the Machine Learning.

We can crudely define the Mathematical Optimization as a field of mathematics in which the task is to find the optimum (minimum or maximum) points of a function $f(x)$. A maximization task for $f(x)$ is the equivalent of a minimization task for $-f(x)$.

3.7.1 Gradient-Based Optimization

Imagine we have a function $y = f(x)$ where $x \in \mathbb{R}$ and $y \in \mathbb{R}$. This function's derivative is denoted by $f'(x)$ or $\frac{dy}{dx}$ and it is the slope of the function f at the point x . In other words, it shows how the function f behaves when we change the x by a small amount

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x) \quad (3.20)$$

This derivation is useful in the minimization task since it shows up how can we change the x , so the overall value $f(x)$ is improved (reduced) slightly. We know that for a small amount ϵ the following is true

$$f(x - \epsilon \text{sign}(f'(x))) < f(x) \quad (3.21)$$

so we can find a minimum point by moving x with small steps in the opposite direction from the sign of the derivative. This method is called Gradient Descent.

A minimum point is any point x that is lower than all its neighbors. The points that the derivative provides no information about the direction of descent, meaning $f'(x) = 0$ are called critical points.

A local minimum point is a point that is lower than all its neighbors, but it is not the lowest point possible for the function. The lowest point of a function is called a global minimum.

For the functions with multiple inputs

$$f : \mathbb{R}^n \rightarrow \mathbb{R} \quad (3.22)$$

we use the partial derivatives. A partial derivative is denoted with $\frac{\partial f}{\partial x_i}$ and it shows the rate of change only for variable x_i .

The gradient of f is a vector containing all the partial derivatives of the function, denoted by $\nabla f(x)$. This vector is pointing to the steepest ascend, and the negative of this vector points directly to the steepest descent.

Using this knowledge, we can devise that for a small step ϵ , the steepest descent at $x^{(1)}$ proposes a new point

$$x^{(2)} = x^{(1)} + \epsilon \nabla f(x) \quad (3.23)$$

where $x^{(2)} < x^{(1)}$.

This small step ϵ that was mentioned previously is called Learning Rate, and it designates how fast the algorithm descends to the minimum.

3.7.2 Stochastic Gradient Descent

The algorithm that drives the process of learning in, arguably, all of Deep Learning approaches is Stochastic Gradient Descent (SGD). This is a specific type of Gradient Descent algorithm. Using a large enough dataset is paramount for a successful generalization. However, using a large dataset is not always feasible from a computational or memory capacity point of view.

The cost function of a learning algorithm is often formulated as a sum of individual error term. For example, the negative conditional log-likelihood can be formulated like

$$J(\theta) = \mathbb{E}_{x,y \sim p_{\text{data}}} L(x, y, \theta) = \frac{1}{m} \sum_{i=1}^m L(x^{(i)}, y^{(i)}, \theta), \quad (3.24)$$

where L is the loss $L(x, y, \theta) = -\log p(y|x; \theta)$.

The gradient of J would be

$$\nabla J(\theta) = \frac{1}{m} \sum_{i=1}^m \nabla L(x^{(i)}, y^{(i)}, \theta) \quad (3.25)$$

The cost of computing this gradient for a dataset with millions of data will take extremely long for each step. The insight of Stochastic Gradient Descent is that the gradient is an expectation [29]. In this way, we can just calculate the gradient for a subsample of the whole population. This sample is called a Minibatch $\mathbb{B} = \{x^{(1)}, \dots, x^{(m')}\}$

Here, we can estimate the gradient as

$$g = \frac{1}{m'} \sum_{i=1}^{m'} \nabla L(x^{(i)}, y^{(i)}, \theta) \quad (3.26)$$

For a small enough m' , the cost of calculating the gradient is low and following this modified gradient will lead to a minimum.

3.8 Summary

Artificial Intelligence (AI) is a very wide research area in Computer Science. The necessity of enabling computers to gather their experiences from raw data rises from failed attempts to hard-code the complicated environments of the real world as a set of formal and logical sentences. This approach to AI is called Machine Learning (ML).

Broadly, we can categorize any ML methods into Unsupervised or Supervised Learning categories. In supervised approaches, we provide the machine with the data and its correct labels and machine is tasked to learn a general representation of the data in a way that optimizes a performance measure. In unsupervised approaches, we allow the machine to experience the data and draw its conclusions about it. These conclusions are usually in the forms of probability distributions of the data.

The current focus of Artificial Intelligence (AI) research is on problems which are easy for humans to do but hard to describe using a set of formal definitions. Problems like speech recognition or object detection are among fore-mentioned problems. The main task of the stereo vision, which is to understand the depth of each object in the scene, fits this description perfectly.

Simple machine learning methods are developed to receive the representations of the data instead of using the raw data. Solving this problem requires processing raw data (images) and extracting features from them in a hierarchical manner and use these features to learn. Deep Learning approaches, especially Deep Neural Networks, are unitized to solve such problems.



CHAPTER 4

PROBLEM, DATA, AND SIMULATION

As it has been discussed in chapter 3, machine learning solutions are devised as an alternative solution to hard-coding knowledge into expert systems. In machine learning approaches, we trust the algorithm to achieve a task by experiencing the data and evaluate its expertise by some performance measure. This is a very formal definition of machine learning. We also discussed possible tasks a machine learning algorithm could undertake, how important it is to choose the right performance measure, and in what ways an algorithm can experience the data.

In this chapter, we will discuss the problem we are facing and the data required for a Deep Learning solution. Data plays a vital role in the machine learning process, and its function is usually overlooked. Not all datasets are equally appealing for machine learning approaches.

It is also essential to analyze the domain of the data, specified to the problem we are attempting to solve with deep learning. We will study a real dataset from our problem domain and analyze the dataset's properties, its creators' approach to data gathering, and its advantages or shortcomings.

Afterwards, we will argue why it was necessary to use a simulated dataset. We will discuss the simulation process, properties of the simulated dataset, and their possible shortcomings. We will also include examples from the simulated dataset. Additionally, any individual interested in obtaining the dataset will be guided on how they can download it.

4.1 Why Is Data Important?

To discuss the importance of the data, we should first be able to define it. Verbally, data can be defined as "facts and statistics collected together for reference or analysis"¹. The origin of this word is Latin, and it is the plural of datum. This creates a certain ambiguity since in science, especially informatics and data science, the word dataset is used frequently instead or in place of data. This would be a plural of a plural, i.e., a set containing data, which is in itself a collection of datum. We overlook this ambiguity and use dataset because it is an established convention.

As important as dataset is for scientific disciplines and research, there is no formal scientific definition for the dataset [76]. This is surprising because, despite the lack of definition, its usage is unproblematic in the scientific community.

Renear et al. [76] define a dataset by its features:

1. **Grouping:** In an informal definition of the word dataset, the consensus is on its structure treated collectively as a unit. From this point of view, words like "set", collection, containment are synonymous with the dataset.
2. **Content:** Most definitions imply that the constituents of a dataset are things of some particular kind — for example, Records, Values, Images, Number, etc.
3. **Relatedness:** Although the datasets are thought as a group of constituents (data points), their content is related in a way that is both beyond the definition of grouping, and the identification of the grouped things as all being of the same general kind of entity.
4. **Purpose:** Other than the obvious objective of recording information, datasets are created with alternative reasons. These alternative reasons might be to prove a hypothesis, or confirmation of the existing hypothesis, etc.

4.1.1 What Makes A Good Dataset For Deep Learning?

As mentioned before at section 3.3.3, a machine learning algorithm is experiencing a dataset and learning its features. Logically, the minimum requirement in a machine learning approach to solving a problem is to have access to a good dataset. But what makes a dataset good?

The following are some of the features required in an effective dataset:

1. **Quantity:** This is one of the most, if not the most, important properties that define a good dataset. How large of a data is necessary for a specific approach is debatable, but the consensus is on having as much data as possible.
2. **Variability:** Although a dataset should contain data from a limited domain, it is still important that the dataset would be a fair representation of its domain's

¹ Definition from Oxford Dictionary.

population. A dataset with enough variation helps the algorithm's learning process and generalization and reduces the chance of over or underfitting.

The fact that the age of big data has made the machine learning much easier shows the importance of data. Goodfellow et al. [29] mention a rule of thumb on how much data is necessary for a supervised approach on deep learning. To achieve acceptable performance, around 5 thousand data point per class is needed, and an algorithm will exceed human accuracy when there are about 10 million data points available for it to experience. Working on databases that have less than this amount is an active research area.

4.2 The Problem

Our research is focused on a specific problem related to Minimally Invasive Surgery (MIS) and the domain of Surgical Navigation Systems. Minimally Invasive Surgery is any surgical operations with the goal of minimizing the number of incisions needed to access the interior of the body (Figure 4.1a). There are many different types of minimally invasive surgeries. Laparoscopy, Endoscopy, Colonoscopy, Hysteroscopy, and Robotic Surgeries are among the examples.

It is of little debate that the popularity of Minimally Invasive Surgeries (MIS) is raising among surgeons [5, 31]. This popularity is directly related to the benefits of Minimally Invasive Surgery (MIS) such as [5]

- Reducing the length of hospitalization
- Faster recovery time
- Less chance of infections
- Less interior and exterior scarring
- Reducing blood loss

However, despite the proven benefits of MIS, a significant amount of operations is still done as open and regular procedures [5]. Researchers are trying to reduce the gap between potential patients and advanced surgical procedures, such as MIS, by enabling the surgeon through the use of technology.

MIS is, by definition, a non-direct approach. Surgeons use many mediums to have access or see the interior of the body (Figure 4.1b). Cameras and screens are a prominent medium in such operations. However, watching through a camera have shortcomings and inconveniences for the surgeon. Some of these shortcomings are

- A narrow field of view
- No sense of touch

- Hindered depth perception

The surgeons usually rely on technology to overcome these difficulties. Image-guided surgery (IGS) and Augmented Reality (AR) have been introduced in literature as solutions for such shortcomings [5, 4].



Figure 4.1: Minimally Invasive Surgery (MIS). (a) The surgeons use small incisions to access the interior cavity. (b) The camera and screens are vital mediums in these surgical procedures.

Surgical Navigation Systems utilizing AR has been a popular choice in operations like neurosurgery but using AR in abdomen surgeries such as Laparoscopy or Endoscopy have been challenging since the target tissue within the abdomen deforms in a non-rigid manner during the surgery [5].

Any attempt of augmenting information into the camera feed requires a certain awareness about the camera itself and the environment that the camera resides. This information can be preoperative Computed Tomography (CT) or Magnetic Resonance (MR) images of the operation target. This is to highlight points of interest in the surgeon's view, such as veins, tumors, etc. Figure 4.2 depicts the information augmented into the view of surgeon.

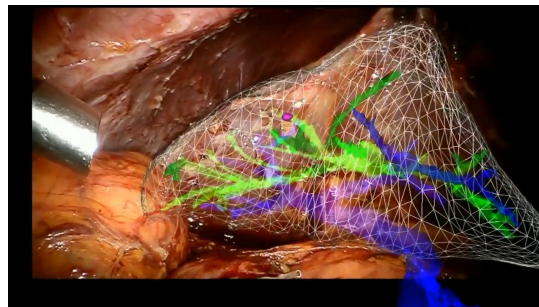


Figure 4.2: Augmented Reality in Minimally Invasive Surgery. (Source: [30])

As it was mentioned before, we use Stereo Cameras and Stereo Vision to reconstruct a three-dimensional representation of the surgical environment. We studied the theoretical background in chapter 2. Figure 4.3 shows the road map to achieve AR from

preoperative information and real-time images of stereo-endoscope. The main focus of this study is on solving the stereo correspondence problem (defined in page 10) and surface reconstruction.

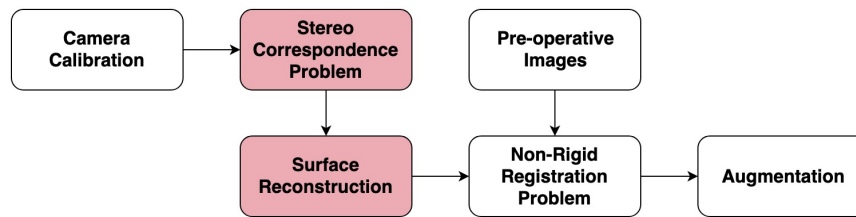


Figure 4.3: Path to achieving Augmented Reality in MIS. Highlighted sections are covered in this research.

4.3 Exploring The Problem's Domain

Although the stereo vision is being utilized in many applications, not all of their problem domains are equally challenging. To understand the challenges, we have examined the real surgical footage that were available at [60, 89, 49, 90, 73, 28, 95, 96]². A list of possible challenges are as follows:

1. **Occlusion by surgical tools:** During a surgical procedure, there are usually multiple tools in between the camera and the scene. These tools occlude portions of the target surface. Additionally, depending on how close they are to the camera, the occlusion will be on a larger surface. There are examples of surgical tools occluding the surface in Figure 4.4.

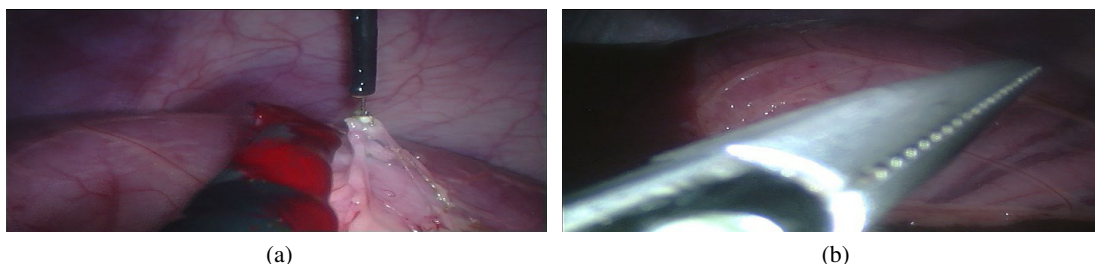


Figure 4.4: Occlusion of the surface by surgical tools. Captured frames from [60] dataset.

2. **Homogeneous Texture:** As mentioned in the section 2.6, we know featureless areas of the images causing mismatches in local methods. Although global methods are addressing this issue much better, it is far from being solved. Figure 4.5 shows some examples from featureless areas in surgical footage. This issue is much greater in this domain when compared with other domains such as Stereo Vision in Autonomous Driving Vehicles.

² Surgical footage are available at <http://hamlyn.doc.ic.ac.uk/vision/>

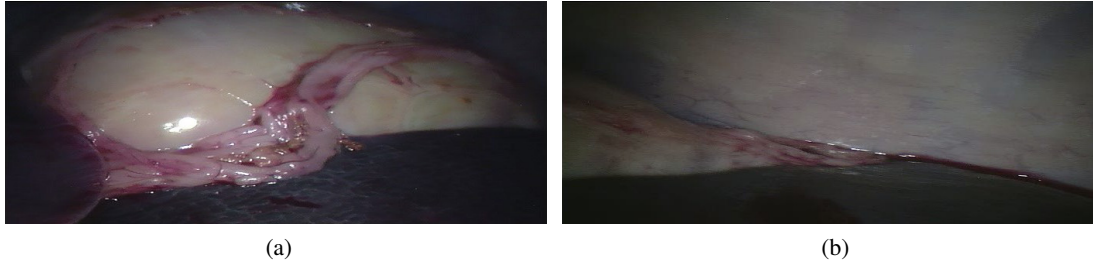


Figure 4.5: Featureless tissues present in the view during MIS. Captured frames from [60] dataset.

3. **Reflection:** Another important issue to address is the reflection. The reflections in this domain are not completely specular, and the wet surface diffuses the reflections to an extent. However, the luminosity of the light source, and illuminating a direct light on the surface usually creates an area with a high amount of reflections. The issue with the reflections is that they will not match on both images. Figure 4.7 illustrates this issue. Not only these large reflections usually cause algorithms to mismatch the immediate reflection area, but they also make it harder for any method relying on optimization to find a minimum for their cost function. Figure 4.6 shows some examples of reflections in this domain.

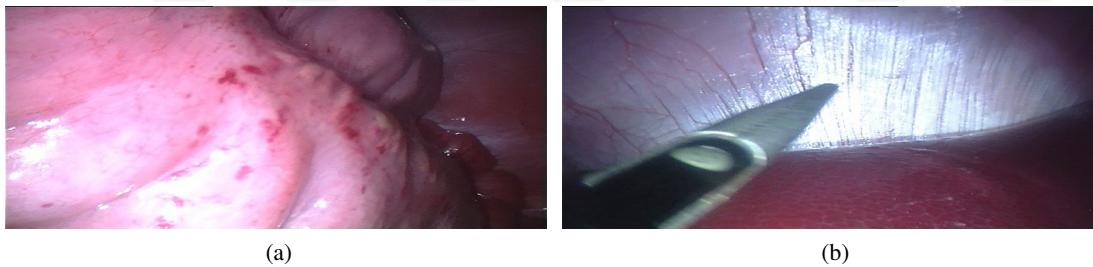


Figure 4.6: Reflections from wet organ surface in MIS. Captured frames from [60] dataset.

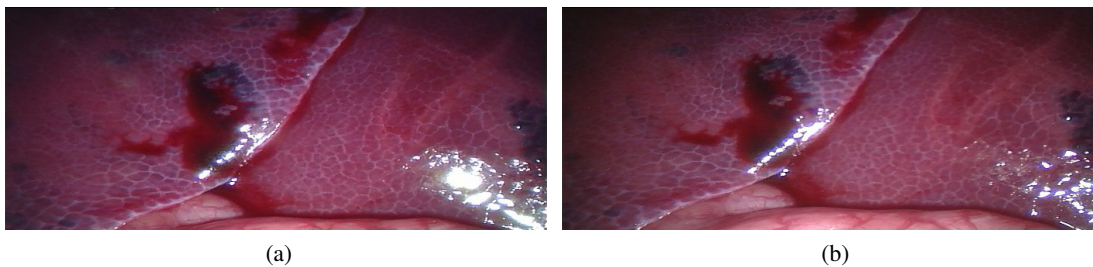


Figure 4.7: Non matching reflections on the surface of the organ from the same frame. (a) Left view. (b) Right view. Captured frames from [60] dataset.

4. **Insufficient Lighting:** Lighting plays an important role in the matching procedure. Without adequate lighting, the images of the scene might not have enough

detail for the matching process. This causes a gradual reduction of luminosity to darker areas (usually closer to the edges of the image) which in turn causes the algorithm's matching error to increase. Figure 4.8 shows some examples.

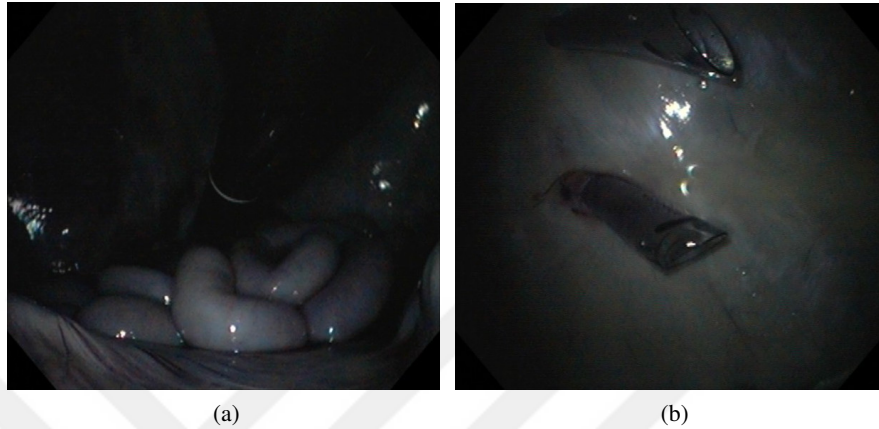


Figure 4.8: Insufficient lighting in MIS. Captured frames from [49] dataset.

- 5. Smoke present in environment:** During MIS, some tools are used to cut and dissect the tissue. For example, Coagulating Hook is a device that uses electricity to cut tissues. The byproduct of using these tools is smoke. The presence of smoke in the camera view during the surgery is common. The smoke is either reducing the visibility over the target surface, causing mismatches by algorithms or completely occluding the surface. Figure 4.9 contains examples of smoke's effects.

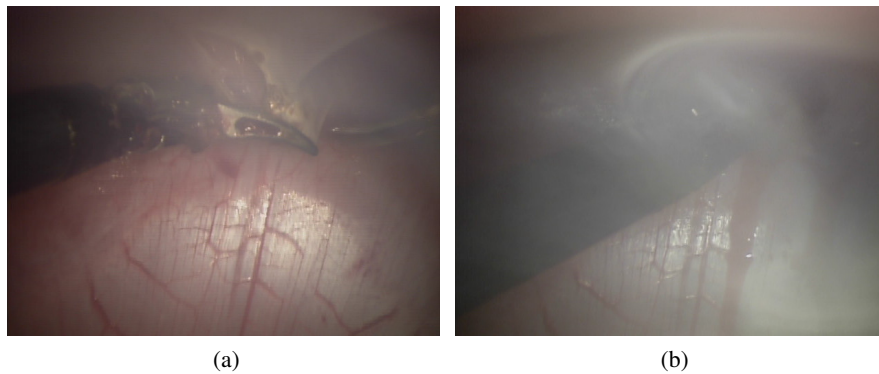


Figure 4.9: Smoke present in the surgical environment. Captured frames from [28] dataset.

- 6. Blood present in environment:** Just like smoke, presence of blood in the surgery environment is highly likely. The blood on any surface indirectly causes sharper reflections, which reduces the capability of the algorithm. Additionally, the blood on the surface will mostly appear as a featureless area, and that is another challenge. Figure 4.10 shows some example of blood in surgical view.

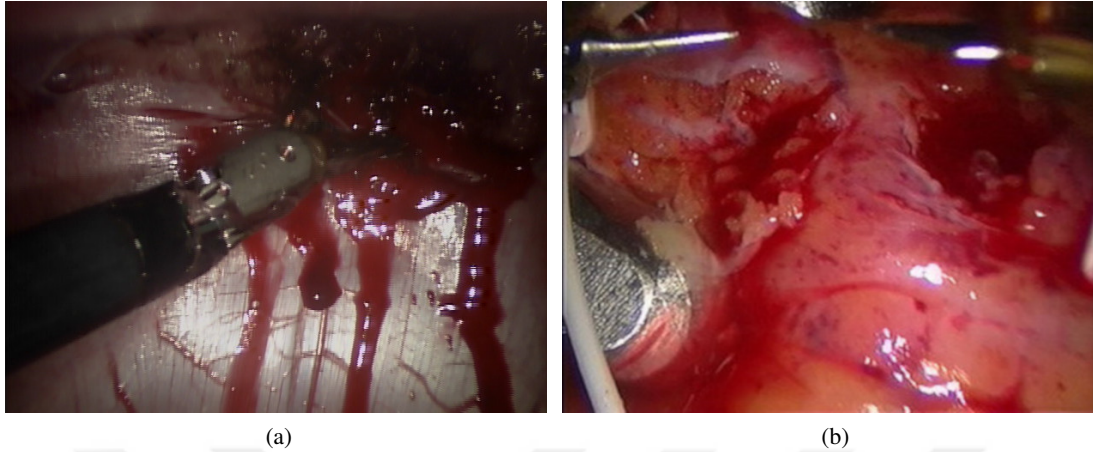


Figure 4.10: Examples of blood on organ surface in MIS. Captured frames from [28, 89] datasets.

7. **Water present in environment:** Another substance that usually appears in the surgical view is water. It is used to clean the surface that is covered in blood or other chunks of tissues. Water is not opaque, but it still bends the light and may cause problems for algorithms. Figure 4.11 shows the use of water during surgery.

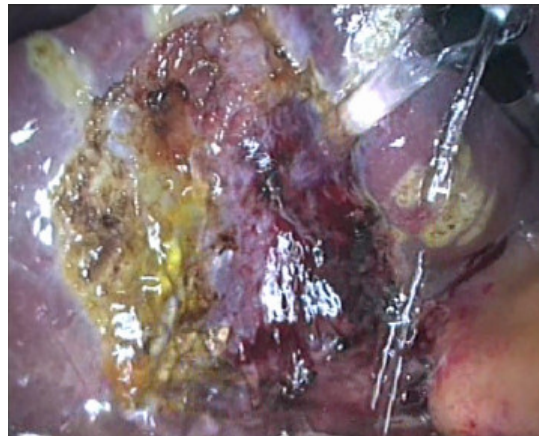


Figure 4.11: Use of water during MIS operation. Captured frame from [28] dataset.

8. **Lens Distortion:** The narrow field of view during the surgery compels the operator to use a lens that shows a wider view. This helps the surgeon viewing more detail, but it also introduces more severe lens distortions. As we know from section 2.3, almost all the algorithms used epipolar restrictions, and this requires the images to become flat and lay down on the same plane. This process will warp the image, and more distortions mean the image will be deformed even more. This potentially causes the algorithm's accuracy to drop, inversely proportional to the distance from the center of the image. Figure 4.12 contains examples of rectified and non-rectified images in this domain.

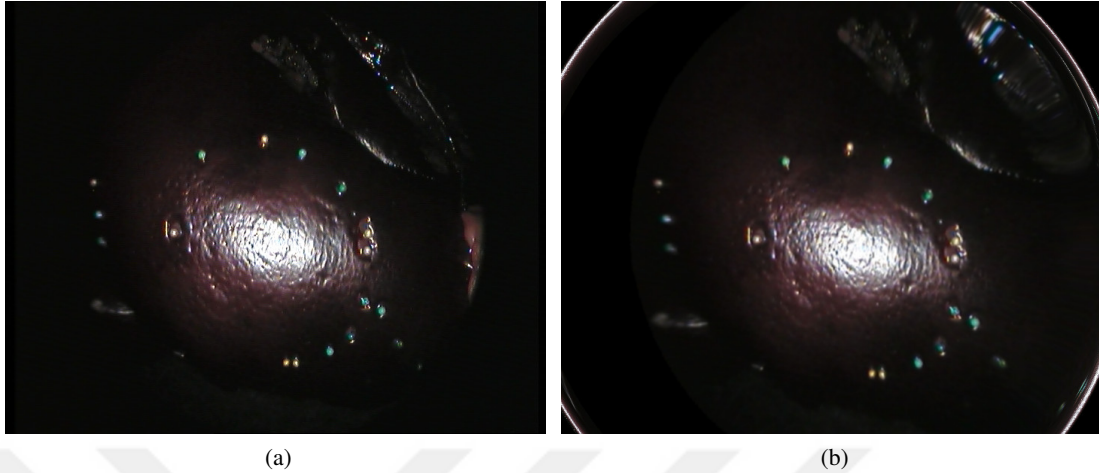


Figure 4.12: Example of lens distortion. (a) Original Image . (b) Rectified image with the distortions visible around the edges of the image . Stereo Images from [55] dataset.

9. **Motion Blur:** An additional problem that may affect the process of stereo matching could be the blurring that happens when objects are moving too fast in front of the camera, or from the camera movement itself. This has the potential to reduce the details of the captured image, and less detail reduces the accuracy of the mapping between the two images. Figure 4.13 shows some images with blurred features due to the motion.

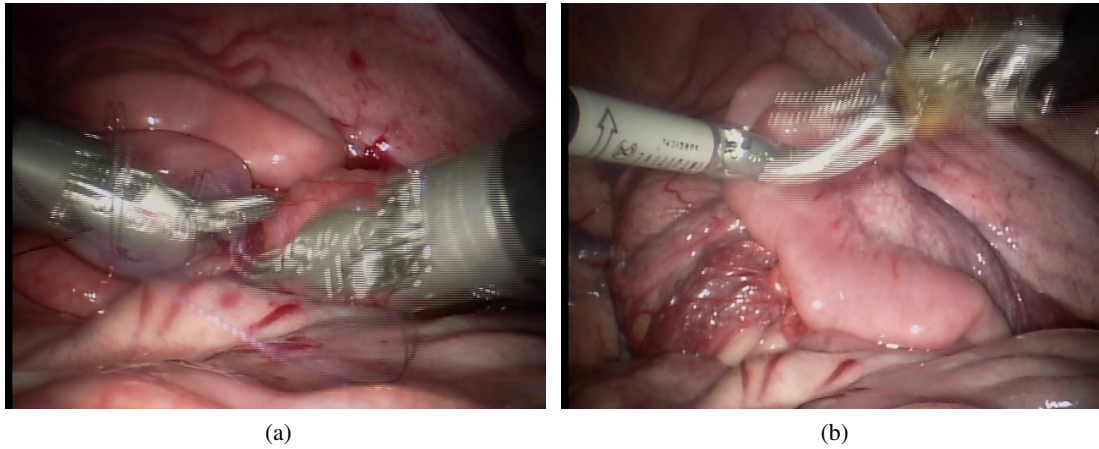


Figure 4.13: Motion blur visible in surgical footage in MIS. Captured frames from [60] dataset.

4.4 Existing Datasets

As mentioned before in chapter 3, any machine learning approach to a problem requires a good dataset. This is because the algorithms will experience the dataset and create its knowledge base. In this chapter, we discussed the importance of the dataset,

Table 4.1: Openly available surgical endoscopic datasets. Adapted from [72]

Surgical Scenario	Organ	Data Format	Truth Information	References
Phantom Organs	Heart	Videos 2 Minutes 45 Seconds	CT Scan	[90, 73]
In-Virto	Porcine Liver, Kidney, Heart, Fatty Tissue	Stereo Images 35 pairs	CT Scan	[55]
In-Vivo	Kidney	Stereo Images $\approx 40,000$ pairs	No Truth	[96]
Phantom Organs	Spleen, Kidney, Liver	Stereo Images 20 pairs	Point Cloud From LIDAR	[72]

and what makes a good dataset. Here we examine some of the existing datasets which provide labeled data. Unfortunately, even if there are databases that contain stereo images from this domain, only a few of them include the truth information [72]. Table 4.1 is a short list of openly available surgical endoscopic datasets. Datasets without the truth information are not useful in any supervised learning algorithms. We will discuss one particular dataset in the following section.

4.4.1 TMI Dataset

The creators of this dataset [55], aimed to enable quantitative and comparative validation for state-of-the-art three-dimensional surface reconstruction in laparoscopic surgery. One of their principal goals is to establish the compatibility of results in different methods. The dataset contains pairs of stereo images from several porcine livers, kidneys, hearts, and fatty tissues³.

The ground truth data was generated from CT images of the objects. The dataset contains images with varying distances from the surface (4cm to 7cm), different angles (0° and 30°), with and without the presence of smoke and blood. Figure 4.14 contains some examples of the dataset's images.

Two stereo endoscope equipment were used in capturing the images, a commercial SD stereo endoscope (image resolution of 720×576 pixels, baseline: $\approx 4mm$, optics: 30°) and a prototype HD stereo endoscope (image resolution of 1920×1080 pixels, baseline: $\approx 3.5mm$, optics: 0°). The cameras are calibrated using a checkerboard pattern, and all the calibration parameters are included in the dataset. Figure 4.15 shows an example from the data gathering process.

Alongside the ground truth information, the authors also included a validation tool which makes evaluating and comparing the result of different algorithms reasonably easy.

³ Dataset is available at <http://opencas.webarchiv.kit.edu/?q=tmidataset>

Unfortunately, a shortcoming of the TMI dataset is its small size. This crucial issue makes the dataset unusable for machine learning approaches.

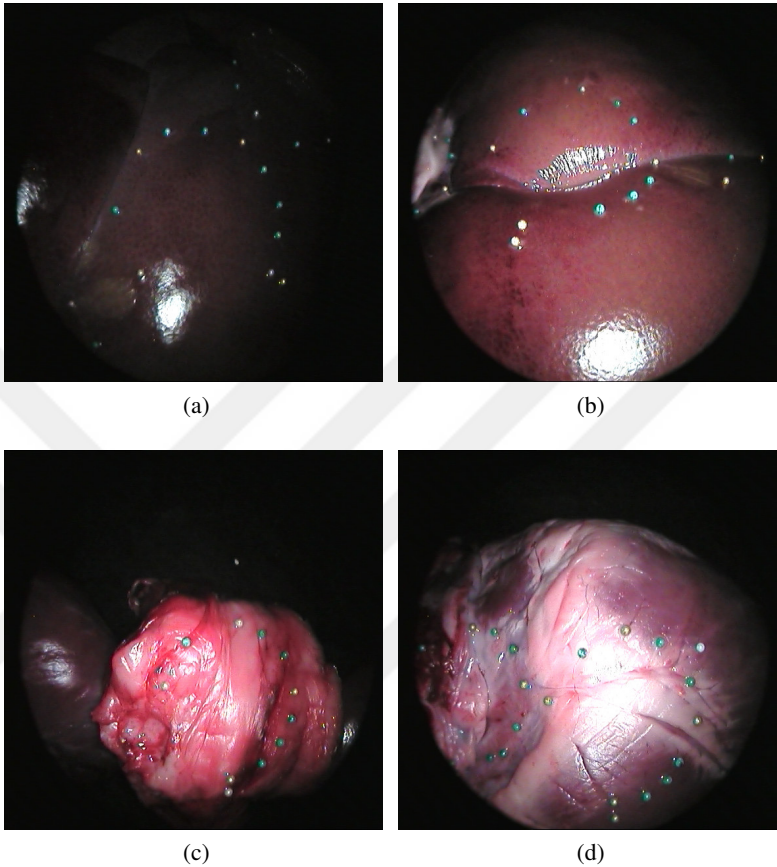


Figure 4.14: Examples from TMI dataset [55].



Figure 4.15: Image acquisition process of TMI dataset. In this image, the smoke is being added to the scene (source: [55]).

This dataset contains a subset of the issues we listed at 4.3. This includes

- Homogeneous Texture
- Reflections
- Insufficient Lighting
- Smoke Present In Environment
- Blood Present In Environment
- Lens Distortion

4.5 Why In-Silico Simulation?

In the past few years, some results indicate that the current algorithms might be limited by the type and amount of labeled training data available to them and not by the models themselves [26]. Moreover, the use of simulations to drive the training process of Deep Learning (by synthesizing the training dataset) has gained attractions in the past few years. So the question is: Why and when is it valid to use synthesized data?

To answer the question above, we need to review three points:

1. As mentioned in chapter 3, the stereo matching (depth perception) is a sort of problem that is intuitive to humans, but it is hard to define in a formal language.
2. We also mentioned in 4.1.1, the amount of data required for successful training of a deep learning method is in thousands, or tens of thousands, of data points.
3. Capturing the ground truth (depth map) of the stereo images is very inconvenient [72]. This is, an even bigger problem since the machine learning approaches need a vast amount of data points.

Knowing these, although a deep learning approach is fitting the problem, the fundamental issue of a "good" training data is remaining. It is sufficient to say that there is a fundamental lack of a "good" dataset for many problems that can be solved by deep learning.

The rapid progress of Computer Graphics research alongside the Computer Vision (CV) research allows the researchers to use synthesized datasets to train and evaluate their models.

4.5.1 Related Research

As mentioned before, this trend of using synthesized data is gaining popularity in recent years. In an article titled "The Reasonable Effectiveness of Synthetic Visual Data", Gaidon et al. [26] state that this trend leads to

“[...] exciting new research directions [...]”

and

“[...] turning the brute-force approach of manual labeling of ground truth and costly sessions of data acquisition, into the generic scientific problem of how to train and test visual/sensorimotor models with synthetic data so that we can ensure that they can ultimately operate in real-world conditions.”

Although this research trend is novel, there are some scientific articles around this subject. They include a wide range from synthesizing data to specific domains [39, 2, 83, 78, 34], to creating a virtual environment that can be used in data generation for a variety of purposes [61]. We will not analyze each of these individual articles but highly recommend follow up with them for anyone interested in this new research trend.

4.5.2 A Hypothesis

We propose an *in-silico* simulation of the training dataset based on real datasets that are publicly available. The central hypothesis of the research is to evaluate the ability of a deep learning solution to stereo correspondence problem, that is trained using an *in-silico* simulated dataset, on an *in-vitro* test dataset. In other words,

1. Whether if it is possible to train a deep learning regression algorithm on simulated data?
2. Whether the algorithm is able to rival other established methods on real data?

The first question is the party answered by the new trend of research we mentioned in the last session. However, to the best of our knowledge, there are no examples of such simulations in the Minimally Invasive Surgery (MIS) domain.

4.6 *In-Silico* Simulation

First, we start with the simulation environment, then move on to simulation parameters and dataset parameters. This dataset is also publicly available.

4.6.1 Simulation Environment

We can separate the simulation process into two stages. The first stage is to create the 3D environment, including all the polygon meshes⁴, locations of light sources,

⁴ In computer graphics terminology, a mesh is a collection of vertices, edges, and faces that describe the shape of a three-dimensional object

and cameras, etc. The second stage is to render the environment using shaders⁵, materials⁶, etc.

Most of the research in this area (some mentioned in 4.5.1), uses a collection of pre-made three-dimensional models, like cars and building, to populate the environment. Our approach to the simulation was to populate the environment with our models. We used Blender⁷ three-dimensional computer graphics software as both modeling and rendering tools (Figure 4.16).

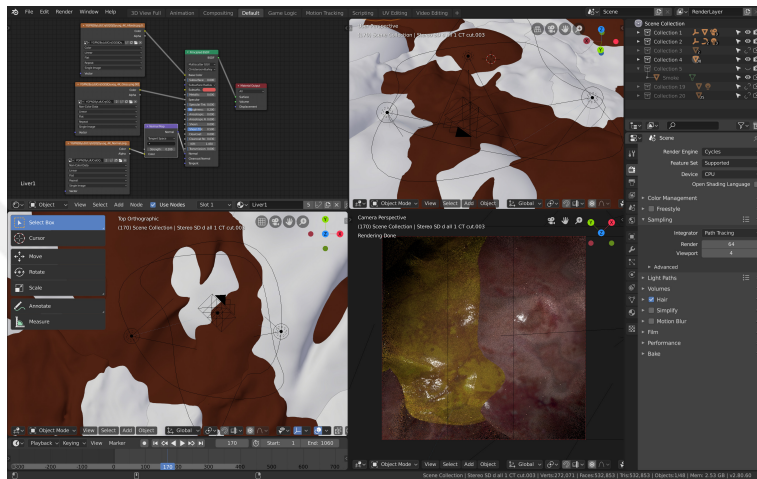


Figure 4.16: Blender three-dimensional computer graphics software as the simulation environment.

Additionally, similar researches usually use a game engine, like Unreal Engine⁸ or Unity Engine⁹, as the rendering engine. The advantage of using such software is that they do not need a huge computational capability. They can render many frames in a second by using a technique known as Rasterization. We used a Ray Tracing¹⁰ Render Engine, named Cycles¹¹, to render the images of the scene. The advantage of a ray tracing engine is that it has the potential of creating more realistic images than any solution relying on Rasterization. But, the downside of using ray tracing is that they are computationally expensive and will take several seconds to render a single frame. This decision was due to the importance of realistic reflections in our simulations.

4.6.2 Simulation Parameters

1. Mesh and modeling

⁵ In computer graphics terminology, a shader is a small program that receives a pixel's coordinate, and returns a color value.

⁶ In computer graphics terminology, the material is a system that defines how the lighting model interacts with the surface of the objects.

⁷ Blender is open-source and available for free at <https://blender.org/>

⁸ Free for any project with revenue of less than 3.000 USD, quarterly. Available at <https://unrealengine.com/>

⁹ Free for any project with revenue or funding of less than 100.000 USD, annually. Available at <https://unity.com/>

¹⁰ A Ray Tracing rendering engine simulated the flow of light rays to create an image.

¹¹ <https://cycles-renderer.org/>

We modeled the three-dimensional mesh of the organ surface in a pseudo-random fashion. This mesh is extending over an area with a set of the stereo camera directly observing the mesh (Figure 4.16). The cameras will scan the mesh, and each frame will be saved.

2. Displacements of the models

We know from chapter 2 that the depth is inversely related to the disparity. We chose the TMI dataset [55] as a visual reference for simulation. To calculate the minimum and maximum amount of mesh displacement in the direction of depth, we need to know what is their minimum and maximum amount of disparity in the TMI dataset's images. However, this dataset does not provide the ground truth information in the form of disparity maps. Since we will be simulating rectified images, this geometry dictates that the minimum disparity would be 0. To calculate the maximum disparity, we used a method called Semi-Global Matching [36] to calculate the disparities for each image in TMI dataset. Figure 4.17 illustrates a single example. By using the same procedure on all images and creating the histogram (Figure 4.19) we decided that the maximum disparity is around 50 pixels.

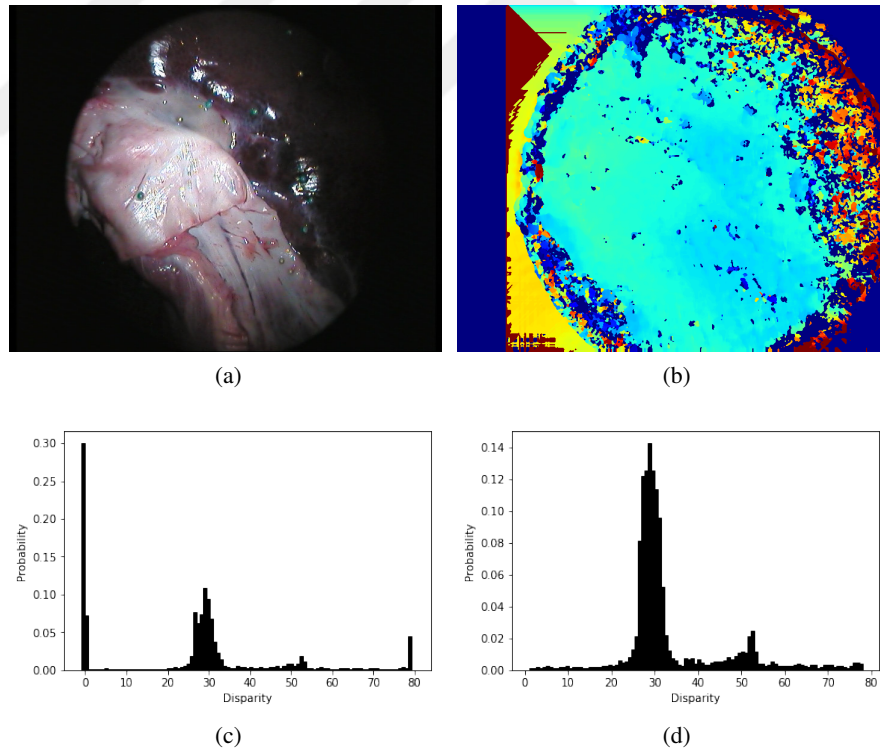


Figure 4.17: Disparity, calculated by Semi-Global Matching [36], and its histogram. **(a)** The image from TMI dataset. **(b)** The resulting disparity map colored for visual appeal. Disparities increase from colder colors to warmer colors. **(c)** The histogram of disparities. There are 4 main distributions within the histogram. The most warm and the most cool parts are outside the circular area, and the distribution around 52 (yellow areas in image) are also outside the circular area. The majority distribution is around 30 (light blue colors). **(d)** The distribution of disparities without the most and the least values.

Additionally, training rectified images will reduce the complexity of the dataset since we do not need to account for different possible geometrical configurations. In other words, since all the different stereo endoscope configurations (camera and system geometry) can be reduced to a rectified geometry, we only focus on the rectified model.

3. Simulating tissues with textures and materials

For our mesh to be realistic, we need to mimic the looks of the tissues using textures. We have created several types of tissue textures using a handful of original textures that all are gather/purchased from online resources.

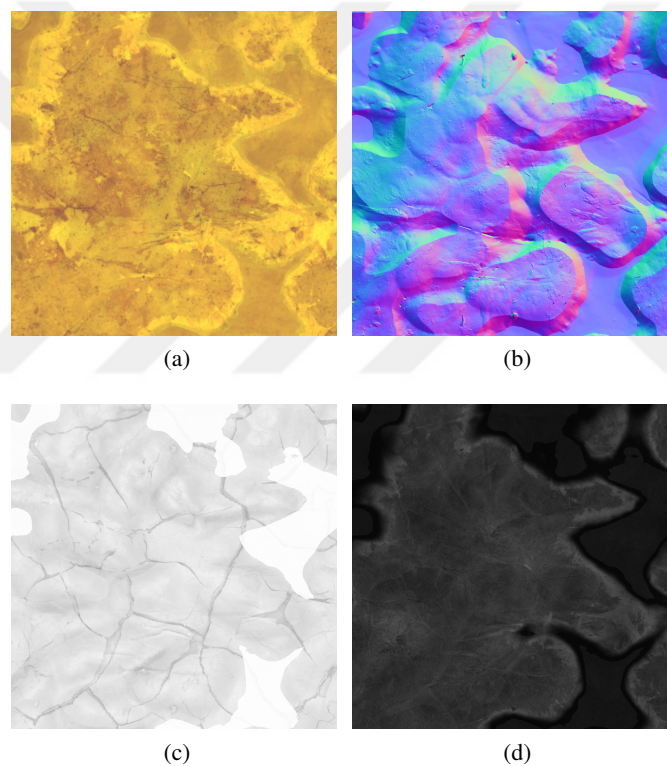


Figure 4.18: An example texture used in the study. **(a)** The color map. **(b)** The normal map. The purpose of the normal map, also know as a bump map, is to fake the bumps on a surface to make it more believable without deforming the original mesh. **(c)** The Ambient Occlusion (AO) map. This map is to create soft shadows. **(d)** The gloss map. This map is to define the roughness of the surface.

Additionally, the textures contain different types of maps designed to make the mesh more realistic. Figure 4.18 contains an example. It is noteworthy that the textures themselves cannot mimic realistic light disbursal, and they need to be accompanied by material systems to become visually realistic. We argue that simulating a vast set of organ textures are not necessary. We assigned all the organs' textures into several categories. Although expanding this library to be more realistic can be useful, we evaluated our set to be adequate.

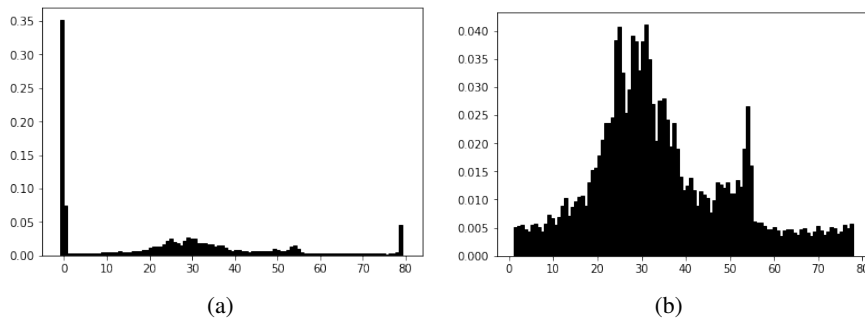


Figure 4.19: Histogram for all of the TMI dataset disparities. (a) The complete histogram. (b) Two central distributions inside the histogram.

4. Simulating lighting

We know from section 4.3 that the reflections from the organs' surface are an issue for conventional matching algorithms. Hence, lighting simulation is important. Alongside using Ray Tracing Engines for more realistic light simulation, we also tried to mimic the level of lighting observed in TMI Dataset [55].

To this aim, we divided the lighting of the environment into three sources. One source is static and perpetually illuminates the center of the scene, and the two additional lightings are dynamically angled. Our dataset has three levels of lighting: low, medium, and high. All of the types have the static light source, but the medium and high levels each have additional dynamic sources added to the scene.

5. Simulating smoke

Another important variable is the smoke simulation. We have already discussed the presence of smoke in the scene. Although defogging is possible as a pre-processing step [54], we have decided that the smoke should be simulated to make it a controllable variable. This decision is because of the varying amount of smoke that can be in the scene at any moment, sometimes covering the surface completely (Figure 4.20).

Initially, the volumetric smoke simulation was attempted in the scene, but the simulation of smoke proved to be extremely expensive. To overcome this, we simulated the smoke separately then augmented the smoke into the surgical view.

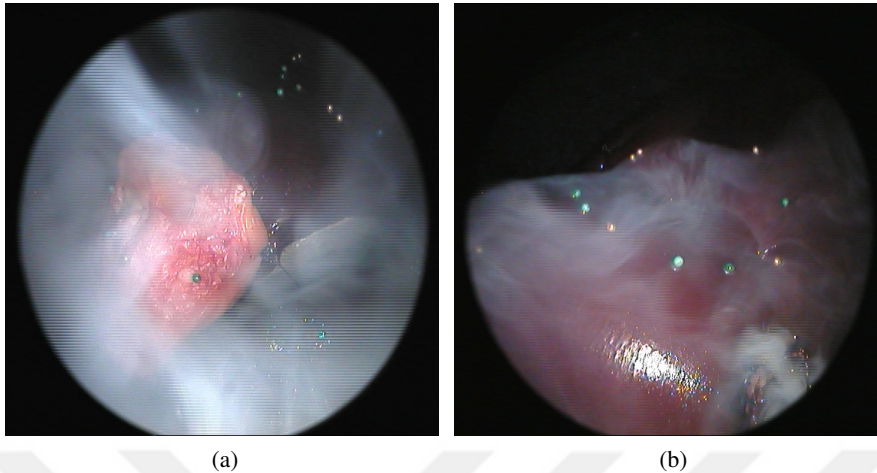


Figure 4.20: Smoke examples from TMI dataset [55].

We tried to adjust the characteristics of the smoke, so the simulation is close to TMI dataset [55]. As shown in figure 4.20, the smoke in the surgical scene is a consistent stream that disperses into a fog like a plume similar to the smoke rising from a cigarette. We used the smoke simulator of the Blender itself with variables as follow:

- (a) Density: 10^{-4}
- (b) Vorticity: $1/4$
- (c) Temperature Difference: 10^{-2}

Additionally, a turbulence field was added so the smoke would disperse into the environment in a pseudo-random manner.

6. Simulating blood

We decided to skip the simulation of blood for two reasons. First, this decision was made because the essential problem with the blood is that its surface resembles of the featureless regions, and those regions are included in simulations. Second, we skipped the simulation in favor of time. It is sufficient to say that the simulation of the featureless sections of the surface will compensate for blood.

7. Output image properties

The output images of the simulations would be with the spatial resolution of 512×512 pixels. This is by design since the images with the spatial resolution that is a power of two are favorable when they are to be fed into a deep neural network. This idea, alongside the solution on nonsquare input to the neural network, would be explained further in section 5.2.4.

Additionally, the output images are not compressed. The changes to the images during the compression process may seem insignificant to the human eye, but they may affect the accuracy of the model. Hence, the stereo images are all in uncompressed PNG (Portable Network Graphics) format. Figure 4.21 contains examples.

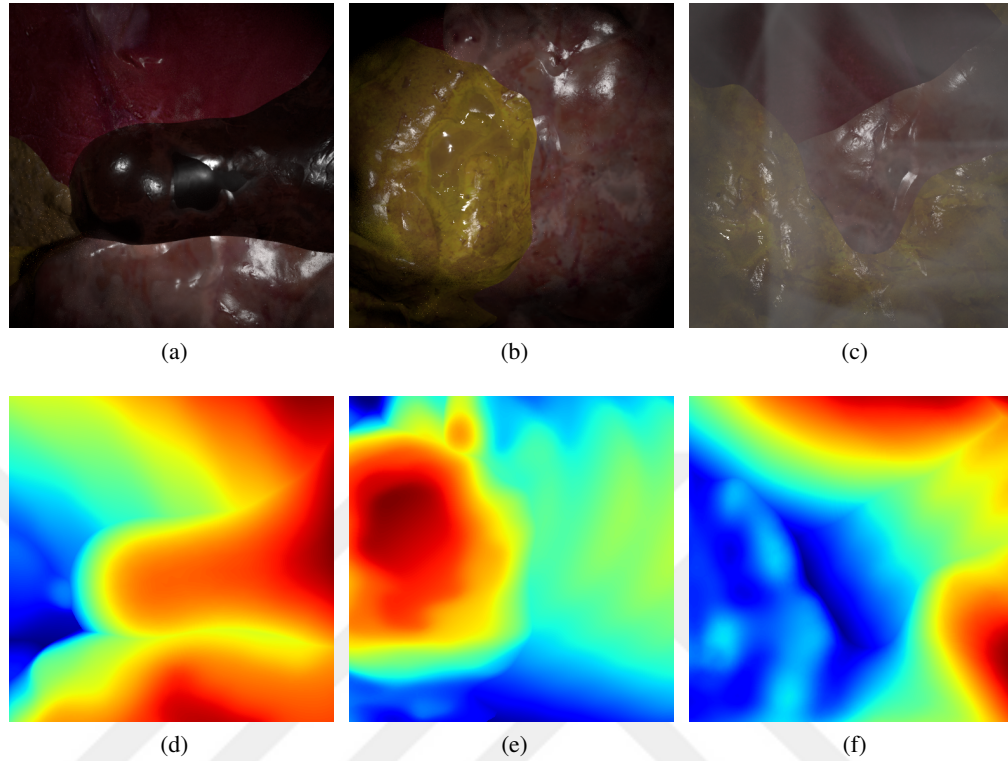


Figure 4.21: Examples from simulations. **(a)(b)(c)** Simulated surfaces. **(d)(e)(f)** Their respective ground truth, color coded for better visual appeal.

Although the disparity output format can be a conventional image, this would be limiting. The disparity values of the simulation would be in floating point numbers, but the conventional images consist of integer numbers. This would result in a substantial loss of accuracy. Because of this, an image format has been devised to contain pixels with floating point values. This is known as PFM (Portable Float Map) format, which can be single-channel (gray-scale images) or three-channel (color-images), but the single-channel type is being used to store disparity values. Since this format is not conventional, it will not be displayed on most computers but converting them to conventional formats (to be human friendly) is easy.

4.6.3 Dataset Parameters

Our dataset contains 9504 stereo images with their respected disparities included, summing up to around 25 GB of data (about 14 GB in compressed format). There are three types of partitions on this dataset.

First, the necessary partition is to divide the dataset into a training set and a validation set. The training set contains 90% of the dataset, where the remaining 10% is for validation purposes. This type of division is commonly used in Machine Learning and Deep Learning research.

Table 4.2: The partitions that divide the dataset into clusters of similar images.

	Without Smoke	With Smoke
Low Amount of Light	A1, A2, A3 A4, A5, A6	D1, D2, D3 D4, D5, D6
Medium Amount of Light	B1, B2, B3 B4, B5, B6	E1, E2, E3 E4, E5, E6
High Amount of Light	C1, C2, C3 C4, C5, C6	F1, F2, F3 F4, F5, F6

The second partition is to divide the dataset into clusters of similar images. We have three lighting levels with smoke simulation, overall six divisions. These partitions are designated with letters from A to F. Table 4.2 shows these divisions.

The third partition is to divide the dataset into clusters of data which are simulated from the same environment. These partitions are designated with digits from 1 to 6 (Table 4.2).

Figure 4.21 contains some examples of the simulated data. To see more samples or gain access to the complete dataset, visit <https://misview3d.aminzabardast.com>.

4.7 Summary

Two factors are contributing to the success of any deep learning solutions: The adequacy and architecture of the learning algorithm itself, and the dataset it experiences. A good dataset is as important as the algorithm itself, and without one, no solution yields good result. A good dataset is a dataset that is large in quantity and has enough variation that it can be a fair representation of the population it represents.

Like any other form of surgeries, Minimally Invasive Surgeries (MIS) have its shortcomings. These shortcomings are related to non-direct access during MIS. A proposed solution to this problem is to enable the surgeon through the use of Augmented Reality (AR). To achieve AR, a three-dimensional knowledge of the surgical scene is necessary. The problem at hand is to create this virtual representation by extracting depth information from stereo images.

By exploring the real datasets that are available publicly, it is clear that none of these datasets are fit to be used in a supervised approach. This is mostly because of the quantity of the data points, but also lacking the ground truth information. A solution for these issues is to synthesize the dataset. This approach has gained attention in the past few years.

We have synthesized a dataset containing nearly 10,000 data points in which the focus is to create a simulation of the surgical footage. This dataset is designed to have similarities to the in-vitro data from TMI dataset [55].

CHAPTER 5

A DEEP LEARNING SOLUTION, AND RESULTS

Deep Learning was briefly defined in chapter 3 as an approach to Representation Learning in which we create complex representations of the data using simpler representations in a hierarchical manner [29]. This hierarchy can be viewed as a deep and complex graph, hence Deep Learning.

One of the typical Deep Learning Algorithms is Multilayered Perceptron (MLP), also known as Deep Neural Networks. In this chapter, we will start by briefly defining the Deep Feed-Forward Networks, discuss their structure, and how can they learn from experiencing the data. Then we review the different types of Deep Neural Networks like Convolutional Neural Networks (CNN). We will also define a special structure for Deep Neural Networks known as Auto-Encoders, Encoder-Decoder, or Hour Glass Structures.

Afterward, we will present our Deep Learning approach to the problem that was described in chapter 4. As we already know, the description of the problem fits in the categories that Deep Learning is a fitting solution. However, in addition to the multiple different architectures we tried, we will discuss the training process, training environment, and the hyperparameters and their optimization.

In the end, the evaluation results we obtained by this deep learning approach will be presented. We will present two types of results. One would be the result of different structures' training in the validation dataset, and the other would be the result of using the networks, that are trained on simulated data, on the in-vitro dataset.

As we mentioned in section 3.3.2, choosing the right metric to evaluate the performance of methods is important. Hence, we will also introduce a family of metrics, suggested by Scharstein et al. [84], that were used for the evaluation of the results in this research.

5.1 Deep Feed-Forward Networks

Any Machine Learning algorithm is, in its essence, like a mapping from one space, x , to another space, y . Here, this map is denoted by the function f .

$$y = f(x) \quad (5.1)$$

Deep neural networks are called "networks" because they have composed form different layer connected to each other (output of one layer is the input to the other layer). Another helpful analogy is to imagine each layer as a separate function $f_i, i = 1, \dots, n$. Then, the whole network can be viewed as nested functions

$$y \approx f_n(f_{n-1}(\dots f_1(x))) \quad (5.2)$$

The learning process will dictate how the output layer (f_n) should behave. However, the inner layers' behavior is decided by the learning algorithm. These layers in between the output and input are called the hidden layers. To go from one layer to another, a set of units (neurons) calculate a weighted sum of their inputs; then a non-linear function will be applied on the result. Currently, the most favorite non-linear function is called Rectified Linear Unit (ReLU). Tangent hyperbolic function, $\tanh(z)$, and Sigmoid function, $\sigma(x) = 1/1 + e^{-x}$, are some other examples of available non-linearity functions (Figure 5.1).

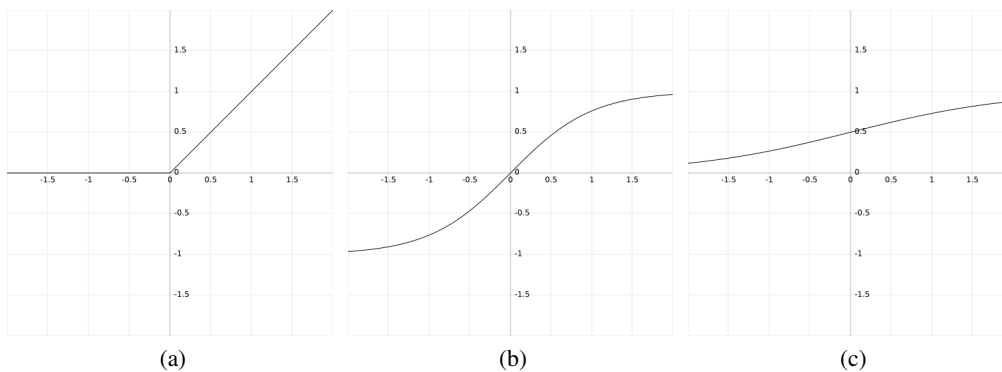


Figure 5.1: Non-linearity (activation) functions commonly used in neural networks. (a) Rectified Linear Unit (ReLU). (b) Tangent hyperbolic function. (c) Sigmoid function.

One very important point is to understand that these functions, $f_i, i = 1, \dots, n - 1$, have non-linear behavior and without it, the networks would not be able to learn the features of complex datasets successfully.

5.1.1 Computational Graph

The nested functions described in the previous section are called a computational graph. A computational graph is not a novel idea, and it dates back to before there

was any talk of neural networks. To define a computational graph, we need to first define two other related terms: computational problem, and computational process. In his article "Computational graphs and rounding error", Friedrich L. Bauer [6] defines a computational problem as

“[...] a set of m functions

$$f_i(x_1, x_2, \dots, x_n), \quad i = 1, 2, \dots, m \quad (5.3)$$

of n real variables x_1, x_2, \dots, x_n from which m real quantities are obtained.”

He proceeded to define a computational process as

“[...] a tissue of number of functions φ_μ of the form

$$\xi_{i,\mu,0} := \varphi(\xi_{i,\mu,1}, \xi_{i,\mu,2}, \dots, \xi_{i,\mu,s_\mu}) \quad (5.4)$$

where the functions are restricted to some relatively elementary ones, mostly with two arguments, like addition or multiplication, and where some of the ξ_i are the input data, and some others the output data of a given computational problem.”

Such a computational process can be defined by a directed graph called computational graph (Figure 5.2). A Deep Feed-Forward Network is essentially a vast and complex computational graph. What makes an Artificial Neural Network (ANN) different than a Computational Graph is the learning through Back Propagation.

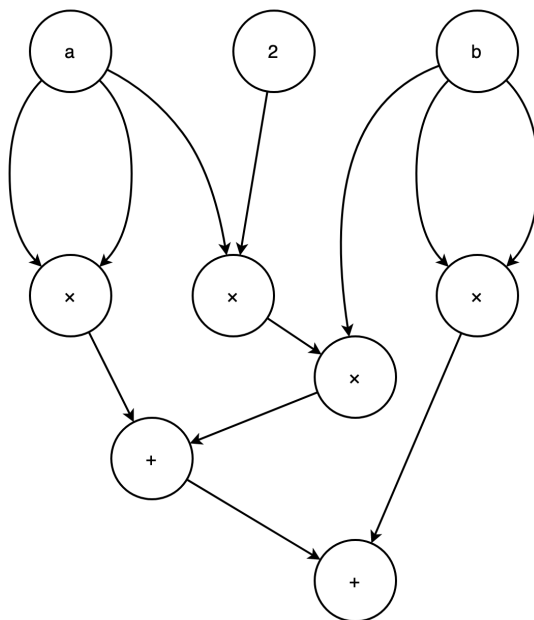


Figure 5.2: An example computational graph for $a^2 + 2ab + b^2$. There can be several computational graphs for a computational problem.

5.1.2 Back Propagation

The learning process in an artificial neural network is a simple stochastic gradient descent which was reviewed in section 3.7.2. As long as each layer of the network is a smooth function of its inputs and its internal weights [46]; the gradients can be calculated using a procedure known as Back Propagation. In the Back Propagation, the chain rule of derivatives is used to calculate the amount of change should be applied to each weights so the overall result is improved (the equivalent of reduced error amount).

Back Propagation is a simple, practical use of the chain rule of derivatives. The only important knowledge is that the gradient of each layer can be calculated by working backward from the output to the input (Figure 5.3b). Once the gradients are calculated, each layer of weights will be updated in the direction of the opposite direction of the gradient. Then the process repeats until we diverge into a fair enough optimum point.

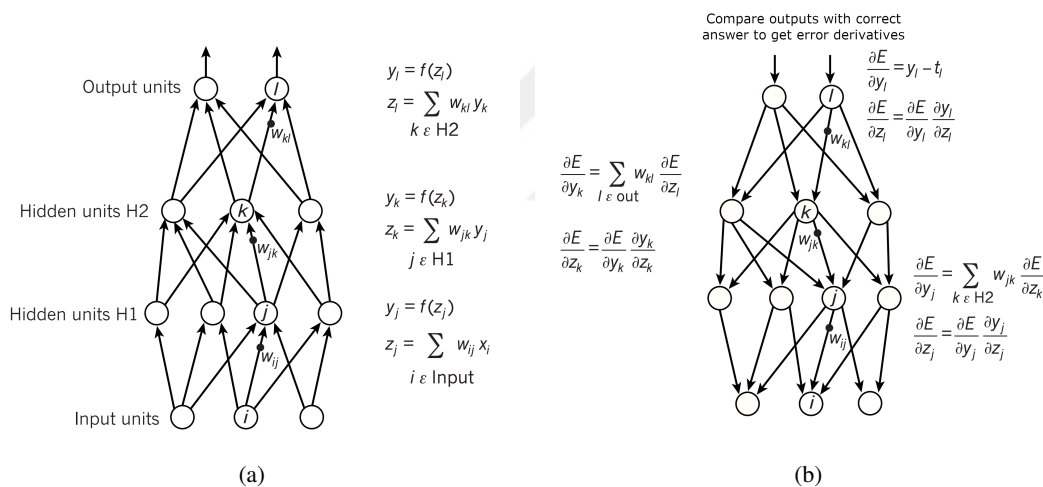


Figure 5.3: Feed Forward and Back Propagation on a simple Neural Network (source: [46]). **(a)** The feed-forward pass within a network with two hidden layers. Output of each layer is the weighted sum of its input values, W , that goes through an activation function, f . **(b)** The back propagation calculations. At each hidden layer we compute the error derivative with respect to the output of each unit, which is a weighted sum of the error derivatives with respect to the total inputs to the units in the layer above. We then convert the error derivative with respect to the output into the error derivative with respect to the input by multiplying it by the gradient of $f(z)$.

5.1.3 Convolutional Neural Networks

Convolutional Neural Network is a special type of Neural Networks designed to accept inputs in the form of tensors¹. For example, language input as one-dimensional

¹ Simply put; Tensors are mathematical objects that are mostly thought of as a generalized matrix. A zero-dimensional tensor would be a scalar; a one-dimensional tensor is a vector, a two-dimensional tensor is a matrix.

tensors, images as two-dimensional tensors, etc. But the most important features of a Convolutional Neural Network are two specialized types of hidden layers. These layers are convolutional and pooling layer.

Convolution Operator

Convolution operator is an operation on two functions. This is like a weighted sum on values of the function f , where the weights themselves are generated by another function g . The function g is generally known as a kernel. The convolution operator is denoted by $*$.

$$s(t) = (f * g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t - a) \quad (5.5)$$

On a continuous form, this sum would be an integral,

$$s(t) = \int f(a)g(t - a)da \quad (5.6)$$

Additionally, we often use convolutions over more than one axis at a time. In that case, if we use an image I as our input, which is two-dimensional, we probably also want to use a two-dimensional kernel K

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \quad (5.7)$$

Convolutional Layer

A convolutional layer is a special layer that uses a convolution operator in place of simple matrix multiplications (Figure 5.4). Using this specialized layer can improve the algorithm in several ways.

One of the most important improvements is that a convolutional neural network is Invariant to Translations. This is especially useful when the input types are multi-dimensional, like images.

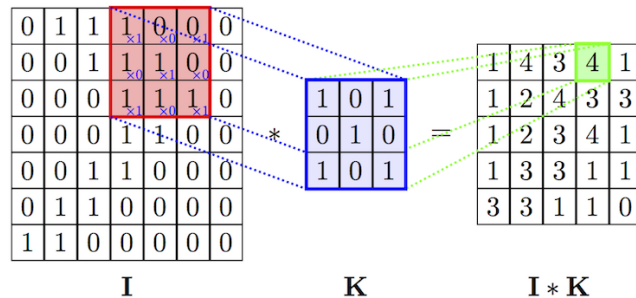


Figure 5.4: Example of Convolution Operator in a Convolutional Layer.

Pooling Layer

A three-dimensional tensor would be a cube-like structure populated by numbers.

Convolutional layers are very good for extracting features, but the feature space they create is not small enough to be meaningful. In other words, the extracted representations should be smaller (in dimensions) than the input data. This is because representations' dimensionality should naturally be smaller than the raw data. Otherwise, they are either as complex as the raw data, or many of them are not useful. This can be viewed as an information reduction.

To achieve this goal, Pooling Layers are introduced. A pooling layer replaces the output of a convolutional layer at a certain location with a summary statistic of the nearby outputs (Figure 5.5). For example, the max pooling [102] layer returns the maximum output within a rectangular vicinity. Other pooling layers can return an average or a weighted average of the vicinity.

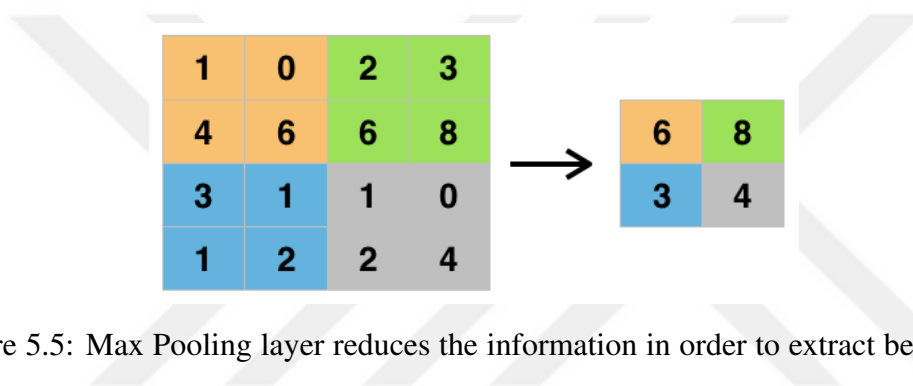


Figure 5.5: Max Pooling layer reduces the information in order to extract better features.

In a typical Convolutional Neural Network (CNN), every convolutional layer is followed by a pooling layer until the feature space is long, and the dimension of the input is smaller. For example, a 128×128 image with color-channels (3) will convert to a smaller but deeper representation like a 4×4 representation with 1024 channels (feature space size). Figure 5.6 depicts a Convolutional Neural Network with the aforementioned structure.

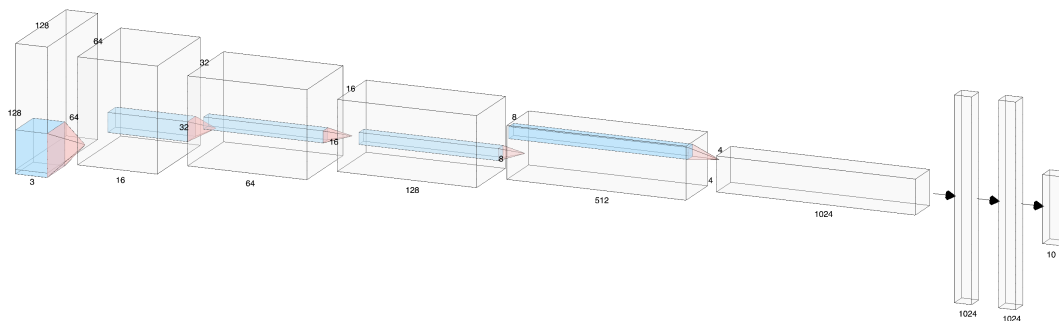


Figure 5.6: Example of a convolutional neural network structure. The input of network is a color image with spatial dimensions of 128×128 (a tensor with size $128 \times 128 \times 3$). The network creates a representation of the image with the size of $4 \times 4 \times 1024$, then inputs the image to a Fully Connected Neural Network for classification.

5.1.4 Auto-Encoders

Auto-Encoders are special neural network structures that are trained to copy their input to their outputs. These structures have a hidden layer, h , that describes a code used to represent the input. These networks can be decomposed into components. An encoder component

$$h = f(x) \tag{5.8}$$

and a decoder component that is used to produce a representation

$$r = g(h) \tag{5.9}$$

As mentioned before, Auto-Encoders can copy their input to their output, but a perfect copy like $g(f(x)) = x$ is not much useful. Figure 5.7 show this general structure. For this reason, Auto-Encoders are designed in a way that they are unable to exactly copy the input to the output.

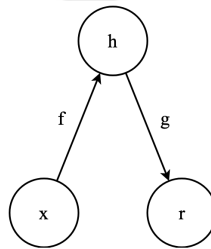


Figure 5.7: General Structure of an Auto-Encoder.

Undercomplete Auto-Encoders

To extract useful information using an Auto-Encoder, we can constraint the encoded representation, h , to have a smaller dimension than the input, x , and output r (Figure 5.8). Such an Auto-Encoder in which the encoded representation has a smaller dimension than the input is called Undercomplete. These structures are also known as hour-glass structures because of the bottleneck in the middle of the structure (Figure 5.8). This constraint forces the Network to prioritize the feature learning only to keep the necessary features from the input.

Choosing the right capacity for the encoded dimension is important because if the capacity is too low, the network will not learn good features and if the capacity is high, then the network learns to copy its input to the output. Of course, the complexity of the data is another important factor in the end result.

The encoder and the decoder parts of the structures can be constructed using convolutional layers, which makes them useful for computer vision purposes.

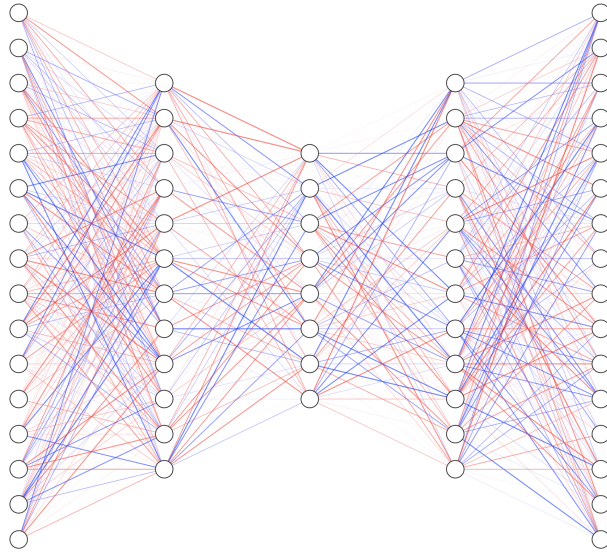


Figure 5.8: An undercomplete Autoencoder. The smaller dimensionality of the encoded representation give an hourglass structure to the whole network.

The researchers have found very useful applications for auto-encoders. Noh et al. [66] have used an Auto-Encoder with convolutional and pooling layers to achieve semantic segmentation². Other researchers have looked into the applications of auto-encoders in denoising [101] and dimensionality reduction [35]. There is also some research around creating super-resolution images [43, 21] using auto-encoders.

There is some related research about using deep neural networks, specially auto-encoders to solve the stereo correspondence problem. Žbontar and LeCun [99] used deep neural networks to calculate the matching cost function for local stereo matching algorithms. This methods still need to be accompanied by a stereo matching algorithm. A similar approach has been attempted by Luo et al. [53]. Pang et al. [69] have attempted to find an end-to-end solution to the stereo correspondence problem using Auto-Encoders. A similar approach has also been attempted by Chang and Chen [16].

Finally, all of the aforementioned related research's datasets are unrelated to the Minimally Invasive Surgeries (MIS) or Medical Fields. Most of the research is around Autonomous Driving Vehicles. We assume this is because of the good datasets available for this domain. Additionally, Autonomous Driving Vehicles research has gathered attention in recent years.

² In computer vision, image segmentation is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify and change the representation of an image into something more meaningful and easier to analyze [87].

5.2 The Solutions and Training Process

We also used Auto-Encoder structures for a solution to the main problem of this research. This is due to the fact that Auto-Encoders solutions are end-to-end, so there is no need for additional algorithms for stereo matching. Moreover, the related research mentioned in the previous section, and the benchmarks mentions in chapter 1 are showing that the performance and accuracy of the Deep Learning approaches are much better than any other methods.

5.2.1 Type S Network: The Simple Solution

The simplest solution is to use a pure Auto-Encoder with convolutional, deconvolutional, layers. We have not used the pooling layers and replaced them with convolutional layers with strides of 2. This means that the kernel slides two pixels at a time. This essentially has the same effect as using a pooling layer after convolutional layer.

The network structure is shown in Table 5.1. The first part of the table is the structure of the Encoder part of the network. The second part of the table is the structure of the decoder part of the network. To be able to input two images into the network, we attach the left and right images (both with 512×512 spacial dimensions, and 3 color channels) to create the input with the size of $512 \times 512 \times 6$ (Table 5.1). The representation created by the Encoder has the dimension size of $8 \times 8 \times 1024$.

The purpose of the Type S Network is to learn all the representation by itself, hence the simple solution.

5.2.2 Type C Network: The Correlational Layer Solution

This solution includes a correlational layer, which provides extra information about the images, which is their correlations. We adapted this layer from Dosovitskiy et al. [22]. The authors of this article created a network that takes two constitutive frames and calculates the optical flow. Although the core problems of calculating the optical flow and the stereo matching problem are different, both networks rely on two frames as inputs. Additionally, the idea of the correlational layer has the same use in both networks.

The correlation is calculated for two patches, one in the vicinity of x_1 from the left image, and the other in the vicinity of x_2 from the right image, using the formula

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k, k] \times [-k, k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle \quad (5.10)$$

for a square shaped patch with size $K := 2k + 1$. The difference between the convolutional layer and this correlational layer is that in the convolutional layer, we convolve with a kernel, but in the correlational layer, we convolve the data with other data. Also, this layer has no trainable parameters.

³ Kernel Size

Table5.1: Network Type S structure guide.

Layer	K ³	S ⁴	Channels I/O ⁵	I/O	Input Layer
conv_1	7	2	6 / 64	512 × 512 / 256 × 256	left+right
conv_2	5	2	64 / 128	256 × 256 / 128 × 128	conv_1
conv_3_1	5	2	128 / 256	128 × 128 / 64 × 64	conv_2
conv_3_2	3	1	256 / 256	64 × 64 / 64 × 64	conv_3_1
conv_4_1	3	2	256 / 512	64 × 64 / 32 × 32	conv_3_2
conv_4_2	3	1	512 / 512	32 × 32 / 32 × 32	conv_4_1
conv_5_1	3	2	512 / 512	32 × 32 / 16 × 16	conv_4_2
conv_5_2	3	1	512 / 512	16 × 16 / 16 × 16	conv_5_1
conv_6 ⁶	3	2	512/1024	16 × 16 / 8 × 8	conv_5_2
pr_6	3	2	1024 / 1	8 × 8 / 16 × 16	conv_6
deconv_5	3	2	1024 / 512	8 × 8 / 16 × 16	conv_6
pr_5	3	2	1025 / 1	16 × 16 / 32 × 32	deconv_5+pr_6+conv_5_1
deconv_4	3	2	1025 / 256	16 × 16 / 32 × 32	deconv_5+pr_6+conv_5_1
pr_4	3	2	513 / 1	32 × 32 / 64 × 64	deconv_4+pr_5+conv_4_1
deconv_3	3	2	513 / 128	32 × 32 / 64 × 64	deconv_4+pr_5+conv_4_1
pr_3	3	2	257 / 1	64 × 64 / 128 × 128	deconv_3+pr_4+conv_3_1
deconv_2	3	2	257 / 64	64 × 64 / 128 × 128	deconv_3+pr_4+conv_3_1
pr_2	3	2	129 / 1	128 × 128 / 256 × 256	deconv_2+pr_3+conv_2
deconv_1	5	2	129 / 32	128 × 128 / 256 × 256	deconv_2+pr_3+conv_2
pr_1	5	2	65 / 1	256 × 256 / 512 × 512	deconv_1+pr_2+conv_1

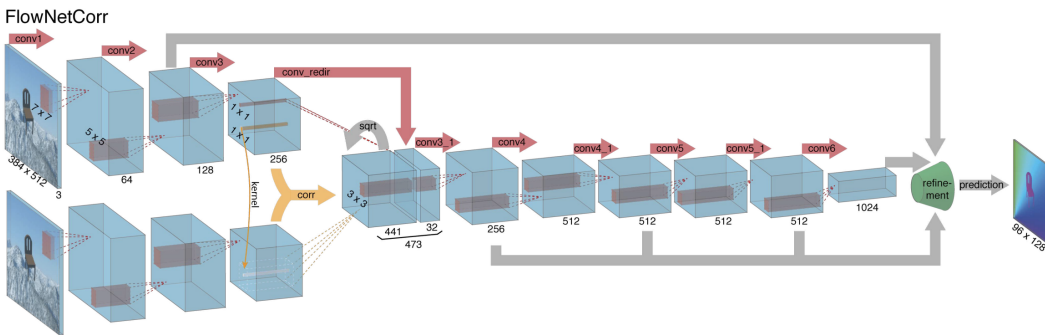


Figure 5.9: The structure of the FlowNet and its correlational layer (source: [22])

⁴ Strides

⁵ Input/Output

⁶ Encoded Representation or Bottleneck Layer

Table5.2: Network Type C structure guide.

Layer	K ⁷	S ⁸	Channels I/O ⁹	I/O	Input Layer
conv_a_1	7	2	3 / 64	512 × 512 / 256 × 256	left
conv_a_2	5	2	64 / 128	256 × 256 / 128 × 128	conv_a_1
conv_a_3	5	2	128 / 256	128 × 128 / 64 × 64	conv_a_2
conv_b_1	7	2	3 / 64	512 × 512 / 256 × 256	right
conv_b_2	5	2	64 / 128	256 × 256 / 128 × 128	conv_b_1
conv_b_3	5	2	128 / 256	128 × 128 / 64 × 64	conv_b_2
cc	-	-	256 / 1681	64 × 64 / 64 × 64	conv_a_2 conv_b_2
conv_redir	1	1	64 / 32	64 × 64 / 64 × 64	conv_a_2
conv_3_1	3	1	1713 / 256	64 × 64 / 64 × 64	cc + conv_redir
conv_4_1	3	2	256 / 512	64 × 64 / 32 × 32	conv_3_1
conv_4_2	3	1	512 / 512	32 × 32 / 32 × 32	conv_4_1
conv_5_1	3	2	512 / 512	32 × 32 / 16 × 16	conv_4_2
conv_5_2	3	1	512 / 512	16 × 16 / 16 × 16	conv_5_1
conv_6 ¹⁰	3	2	512/1024	16 × 16 / 8 × 8	conv_5_2
pr_6	3	2	1024 / 1	8 × 8 / 16 × 16	conv_6
deconv_5	3	2	1024 / 512	8 × 8 / 16 × 16	conv_6
pr_5	3	2	1025 / 1	16 × 16 / 32 × 32	deconv_5+pr_6+conv_5_1
deconv_4	3	2	1025 / 256	16 × 16 / 32 × 32	deconv_5+pr_6+conv_5_1
pr_4	3	2	513 / 1	32 × 32 / 64 × 64	deconv_4+pr_5+conv_4_1
deconv_3	3	2	513 / 128	32 × 32 / 64 × 64	deconv_4+pr_5+conv_4_1
pr_3	3	2	257 / 1	64 × 64 / 128 × 128	deconv_3+pr_4+conv_3_1
deconv_2	3	2	257 / 64	64 × 64 / 128 × 128	deconv_3+pr_4+conv_3_1
pr_2	3	2	129 / 1	128 × 128 / 256 × 256	deconv_2+pr_3+conv_2
deconv_1	5	2	129 / 32	128 × 128 / 256 × 256	deconv_2+pr_3+conv_2
pr_1	5	2	65 / 1	256 × 256 / 512 × 512	deconv_1+pr_2+conv_1

Table 5.2 contains the structure of the Type C Network. The network’s overall structure is similar to Figure 5.9. Two wings will process the left and right images separately. Then their processed representations will be used to calculate the correlational layer. The output of the correlational layer and the left image representation will be concatenated and will go through an Auto-Encode structure.

In contrary to the Type S Network solution, we provide the network with some expert data (correlations between image representations) to help the process.

⁷ Kernel Size

⁸ Strides

⁹ Input/Output

¹⁰ Encoded Representation or Bottleneck Layer

5.2.3 Type CS Network: The Ensemble Solution

In Section 3.4, we discussed that there is a positive correlation between the depth of a Deep Neural Network and its learning capacity. To create a Deeper Network with higher learning capacity, we have created the Type CS Network as an ensemble of the two C and S networks, in the same order. The output of the C network, which is a disparity map (with the size of $512 \times 512 \times 1$), will be used to warp the right image, r , into a pseudo left image, \hat{l} . Afterward, the pseudo left image alongside the disparity map, left, and the right image will be concatenated and used as initial input to the Type S network. In this case, the input of the S network will be in the shape of $512 \times 512 \times 10$. The rest of the networks are the same as the structures in the Tables 5.1 and 5.2.

5.2.4 Network Input Size Issue

Since the input size of the network is 512×512 , one might wonder how can an image with a typical non-squared ration (4:3, 16:10, 16:9, etc.) be fed into the network. The solution is to divide the image, process each chunk independently and then stitch them together. This will not reduce the speed of the network by a considerable amount since all the chunks can be stacked and fed to the network only once (as a Mini-Batch).

Additionally, the division does not need to be a partition, meaning that overlapping areas are allowed to exist. In fact, by increasing the overlapping areas, more samples will be calculated for the pixels around the center. Then, we can achieve a better approximation of the disparity value by averaging all the samples.

In the future section, we explain the result of using the networks to calculate the depth of the images in TMI dataset. We have used this technique to calculate the depth map for the images with the size 720×576 .

These images are divided into overlapping sections (Figure 5.10) and connected after calculations by the networks.

5.2.5 Training Parameters

1. **The Loss Function:** The loss function we chose to train this neural network with is Log-Cosh function

$$L(y, \hat{y}) = \sum_{i=1}^n \log(\cosh(\hat{y}_i - y_i)) \quad (5.11)$$

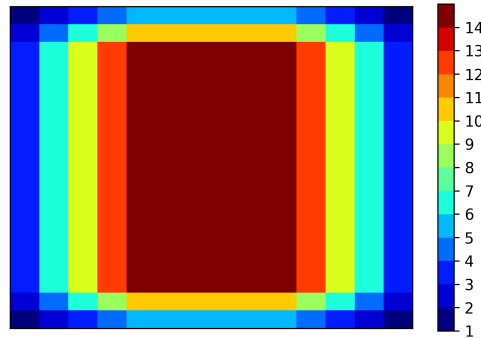


Figure 5.10: The divisions over a TMI dataset image. The concentration of the disparity samples from the network are higher closer to the middle point.

The advantage of this loss function is that it approximates $(x^2)/2$ around small values of x , and $|x| - \log 2$ for large x (Figure 5.11). This means that this loss function works like Mean Squared Error (MSE) but it is not hugely effected by the outliers. Also, the gradient will get smoother closer to the small values. Additionally, This function is twice differentiable everywhere.

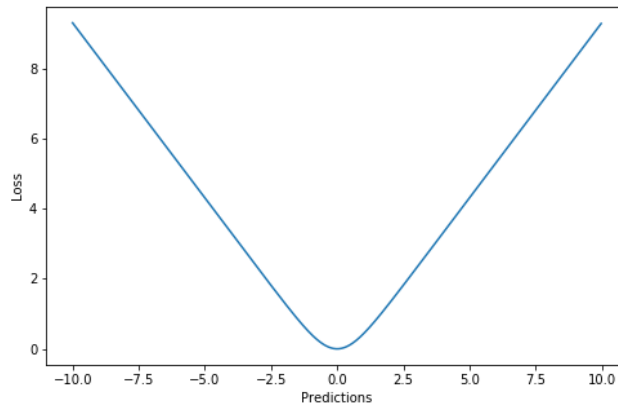


Figure 5.11: The shape of Log-Cosh Loss Function.

2. Epochs and Mini-Batch Size:

We have trained all three models for 100 epochs. Since the data is vast and cannot be loaded completely into the memory, a stochastic approach is attempted. The data is fed to the network in Mini-Batches. Each Mini-Batch will have 15 pairs of images, 30 images overall.

3. Learning Rate:

To find the optimum learning rate, we implemented a call back system that multiplies the learning rate by 0.5 whenever there is no improvement in loss function within three consecutive epochs. Multiple tries showed up that the best learning rate is, to begin with, 10^{-3} and reduce the learning rate to 10^{-4} after around 30 epochs.

4. **Optimizer:**

We used Adam optimizer instead of the classic Stochastic Gradient Descent (SGD). Adam was introduced by Kingma and Lei Ba [44] in 2014. The name Adam is derived from adaptive moment estimation.

Some of the advantages of the Adam optimizer are:

- Memory efficiency.
- Effective against noisy or sparse gradients.
- Well suited for problems that are large in terms of data and/or parameters.
- Straight forward implementation.

In an article titled "An overview of gradient descent optimization algorithms" [82], Ruder presents a comparative study of the optimizations methods for Machine Learning and Deep Learning. In a section titled "Which optimizer to use?", he recommends using Adam:

"Insofar, RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances. [...] its bias-correction helps Adam slightly outperform RMSprop towards the end of optimization as gradients become sparser. Insofar, Adam might be the best overall choice."

5. **Implementation:**

We implemented all three networks using TensorFlow [1] library, version 1.10. The implementation of these networks are available at our repository on GitHub¹¹

6. **Training Environment and Time:**

We trained the networks on a system using Ubuntu 16.04 operation system, and equipped with two Nvidia GeForce GTX 1080 Ti¹² GPUs. The training time, using both GPUs, was around 25 hours for each network.

5.2.6 **Training Process and Performance**

As it is shown in the Figures 5.12 and 5.13, the loss value is being reduced for all the networks during the training and reaches a flat state. This indicated a successful training process.

By comparing the different networks, We see that the Type C Network is more stable than the others. Especially in validation dataset this stable status is visible.

¹¹ Repository available at <https://github.com/>

¹² Around 11.3 Tera FLOPS of computational power for 32bit floating point numbers.

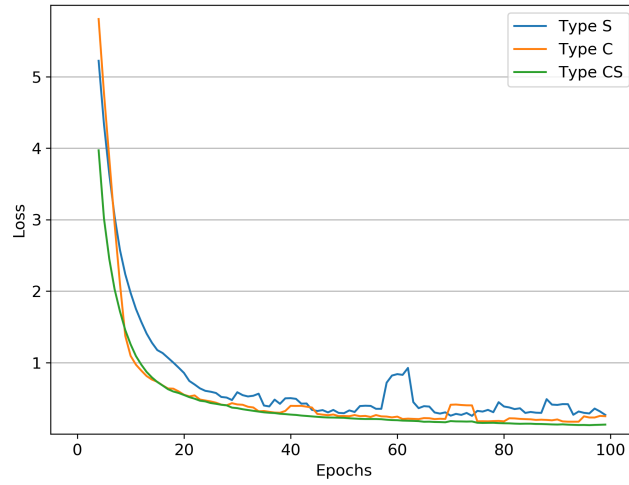


Figure 5.12: The loss per epoch for the training dataset. The stochastic optimization results in a jagged plot, hence we smoothed the plot for better visuals.

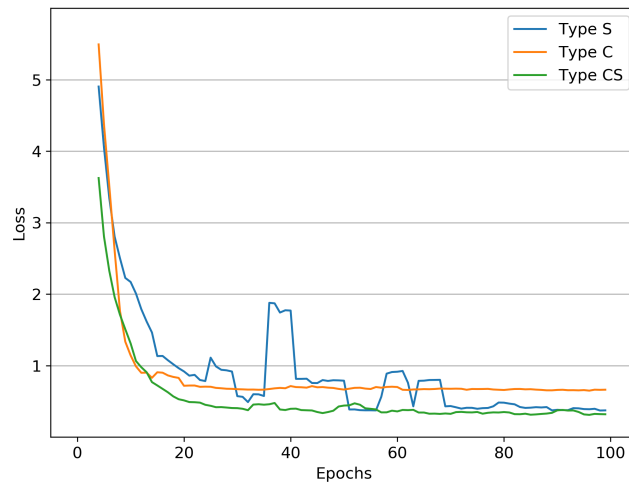
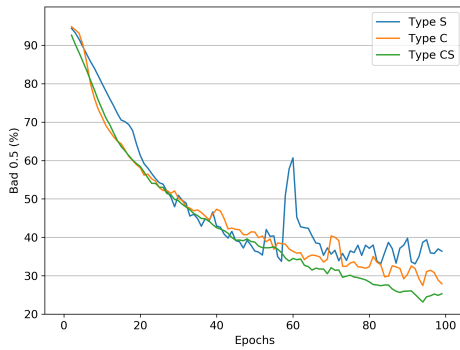


Figure 5.13: The loss per epoch for the validation dataset. The stochastic optimization results in a jagged plot, hence we smoothed the plot for better visuals.

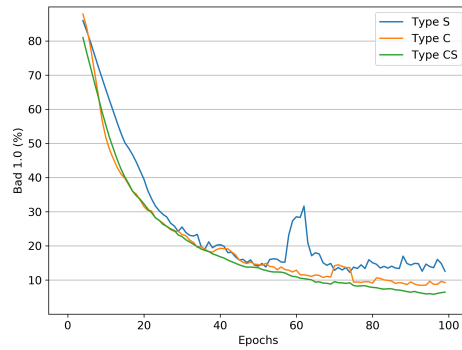
Moreover, we can assess the performance of the networks by several performance metrics. We utilized the performance metrics suggested by Scharstein et al. [84]. Table 5.3 contains these metrics.

Table5.3: The metrics used in this research. suggested by [84].

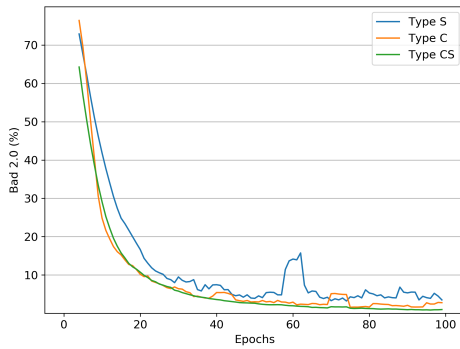
Metrics	Description
Bad 0.5	The percentage of bad pixels with disparity error greater than 0.5 pixels
Bad 1.0	The percentage of bad pixels with disparity error greater than 1.0 pixels
Bad 2.0	The percentage of bad pixels with disparity error greater than 2.0 pixels
Bad 4.0	The percentage of bad pixels with disparity error greater than 4.0 pixels



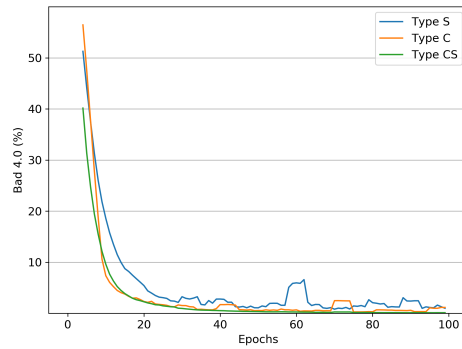
(a)



(b)



(c)



(d)

Figure 5.14: The performance on training dataset, compared using different metrics. **(a)** Bad 0.5 Metric. **(b)** Bad 1.0 Metric. **(c)** Bad 2.0 Metric. **(d)** Bad 4.0 Metric.

On the training dataset (Figure 5.14), all the performance metrics are decreasing during the training process. Type S network seems a bit more chaotic and tagged, but the rest are more steadily decreasing.

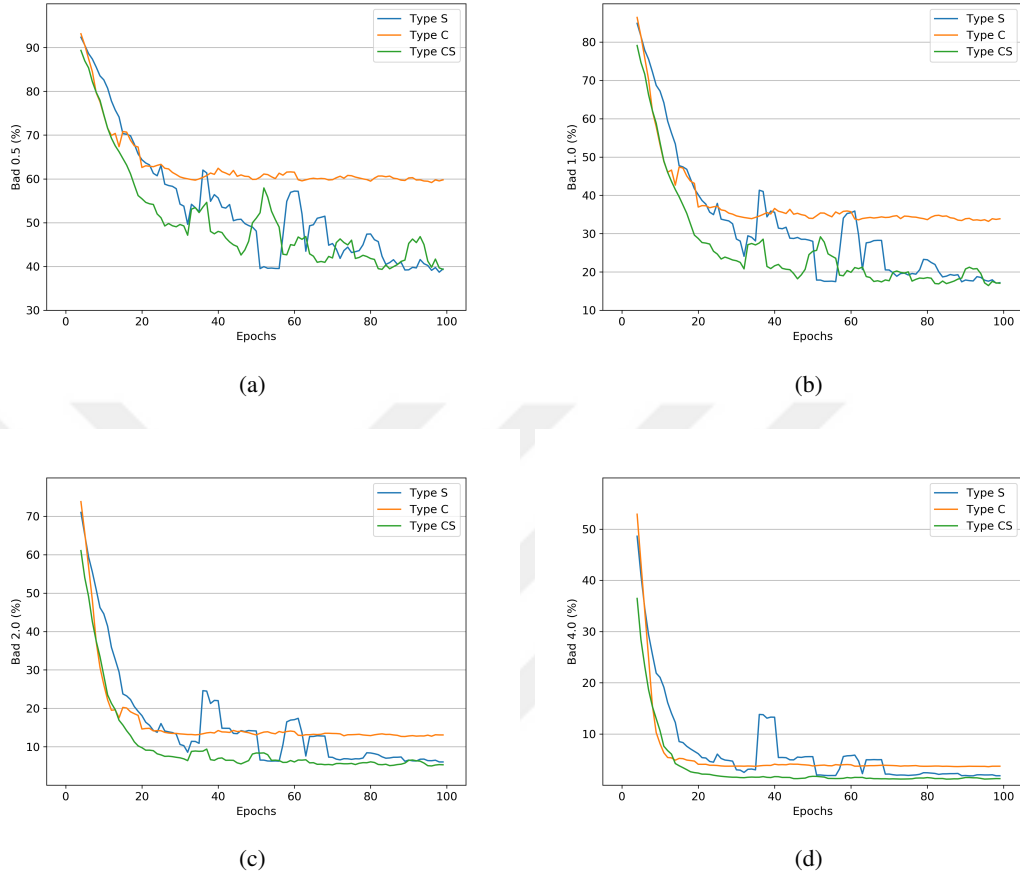


Figure 5.15: The performance on validation dataset, compared using different metrics. **(a)** Bad 0.5 Metric. **(b)** Bad 1.0 Metric. **(c)** Bad 2.0 Metric. **(d)** Bad 4.0 Metric.

On the validation dataset (Figure 5.15), network Type C is decreasing less than the rest and reaches a stable state, while the rest are more chaotic and jagged. This is an indication that the network C is better at generalizing the learning, at least in the current dataset.

5.3 Evaluation and Results

By knowing about the advances in computer graphics and related research in the Deep Learning research community, we hypothesized (section 4.5.2) that it is possible to simulate a dataset by which it is possible to train a neural network. Such a dataset should include enough variation in data and should be large enough in quantity. In the previous section, we explained the training process of three Deep Neural Networks, and their performance on a validation dataset, which is a simulated dataset.

As an initial step, we will compare the Network's result against two other control methods. This will enable us to evaluate the core issue of whether Deep Neural Networks can learn the disparities better than some other methods that are established in the

literature.

To evaluate our hypothesis, we used the three networks, alongside two control methods, to calculate the disparity values for each pair of images within the TMI dataset. As we mentioned in the section 4.4.1, This dataset’s ground truth information are not disparity maps, but rather three-dimensional mesh information extracted from CT images. Fortunately, the creators of the TMI dataset have enabled us with the validation tool¹³ (Figure 5.16) which makes comparisons easier. To achieve this, we need to convert disparity maps to point cloud formats required by the tool. Afterward, we can compare the results of the networks on real data.

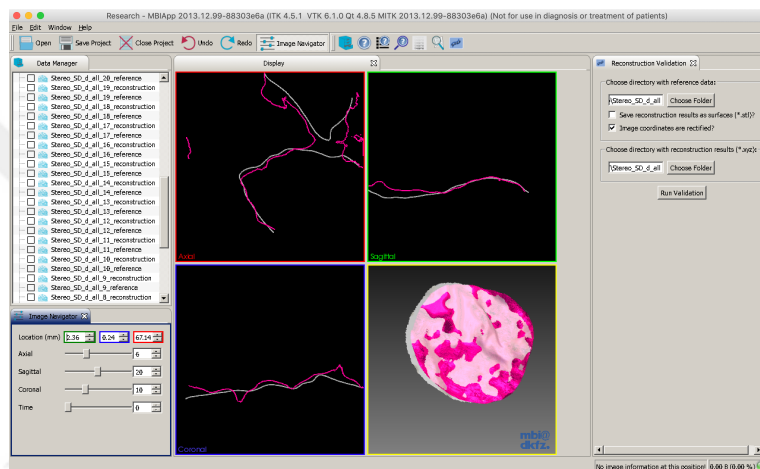


Figure 5.16: The validation tool available for evaluating the results for TMI dataset.

5.3.1 Comparative Study and Control Methods

We chose two methods as control methods in this research. Both of them are fast and reliable for real-time applications. But they are not state-of-the-art. Unfortunately, many of the top performing methods are either not publicly available, or they are not easy to configure and execute.

This is specially true about the other Deep Learning approaches. They are not straight forward in their configuration and execution. They also need long training times which is an additional complication. Overall, using other Deep Learning based methods are a lot more complicated.

We have implemented one of the chosen control methods ourselves, but in favor of reliability and speed, we used the implementations available in OpenCV¹⁴ library.

The methods used as control methods are:

1. Block Matching:

¹³ Available at <http://opencas.webarchiv.kit.edu/?q=node/23>

¹⁴ Version 4.1.0 was used. Available at <https://opencv.org/> and <https://github.com/opencv/opencv/tree/4.1.0/>

This is a completely local method and a simple block matching algorithm. As explained in section 2.6, these algorithms are not as accurate, but they are widely being used because of their simplicity and speed. As the algorithm settings, the number of disparities is 80^{15} , and the window size is 11×11^{16} .

2. Semi-Global Matching:

As the title stated, this method is not a local method and it tries to approximate the disparity using visual clues and mutual information [36]. However, this method is also very fast, and it is very reliable. As the algorithm settings, the number of disparities, and the window size are the same as the previous method. Additionally, There are two variables P_1^{17} and P_2^{18} , that contribute to the smoothness of the output. We used 100 for both of them¹⁹ to output a smoother map than the default values.

5.3.2 Validation Set Results

Before evaluating the TMI dataset, here we compare the performance of the networks in comparison with the control methods, using our performance metrics. We can see in Figure 5.17 that networks overperform the control methods in all of the metrics.

5.3.3 Back Projection To 3D Space

To use the validation tool provided by the creators of the TMI dataset, the Disparity Map should be transformed into a point cloud format. Point clouds are three-dimensional data with the coordinates of the vertices in the three-dimensional environment. The coordinate in use here should be relative to the camera.

We have already mentioned in Chapter 2 that there is a reverse correlation between the disparity values and the depth of the points. This means that the displacement of the object closer to the parallel stereo cameras will be more severe than the object farther away.

This projection matrix is to convert the disparity into the depth values, and it requires c_x, c_y as camera's principle point, f as the focal point, and b as the length of the baseline between two cameras (baseline). u and v are the coordinates of the pixels, and d is disparity of the pixel.

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{b} & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \quad (5.12)$$

¹⁵ numDisparities = 80

¹⁶ blockSize = 11

¹⁷ This parameter controlling the disparity smoothness. P_1 is the penalty on the disparity change by plus or minus one between neighbor pixels.

¹⁸ This parameter controlling the disparity smoothness. P_2 is the penalty on the disparity change by more than one between neighbor pixels.

¹⁹ P1 = 100 and P2 = 100

If we work out the calculations we will reach

$$\Rightarrow X = u - c_x, Y = v - c_y, Z = f, W = -\frac{d}{b} \quad (5.13)$$

and since these values are in homogeneous coordinates (projective coordinates), they should be converted to Cartesian coordinates.

To convert the homogeneous coordinates into Cartesian coordinates, we need to make sure that the last coordinate equals to 1. So we divide all the previous coordinates by the last one to get Cartesian coordinates in three-dimensional space.

$$\Rightarrow X' = -\frac{(u - c_x)b}{d}, Y' = -\frac{(v - c_y)b}{d}, Z' = -\frac{fb}{d} \quad (5.14)$$

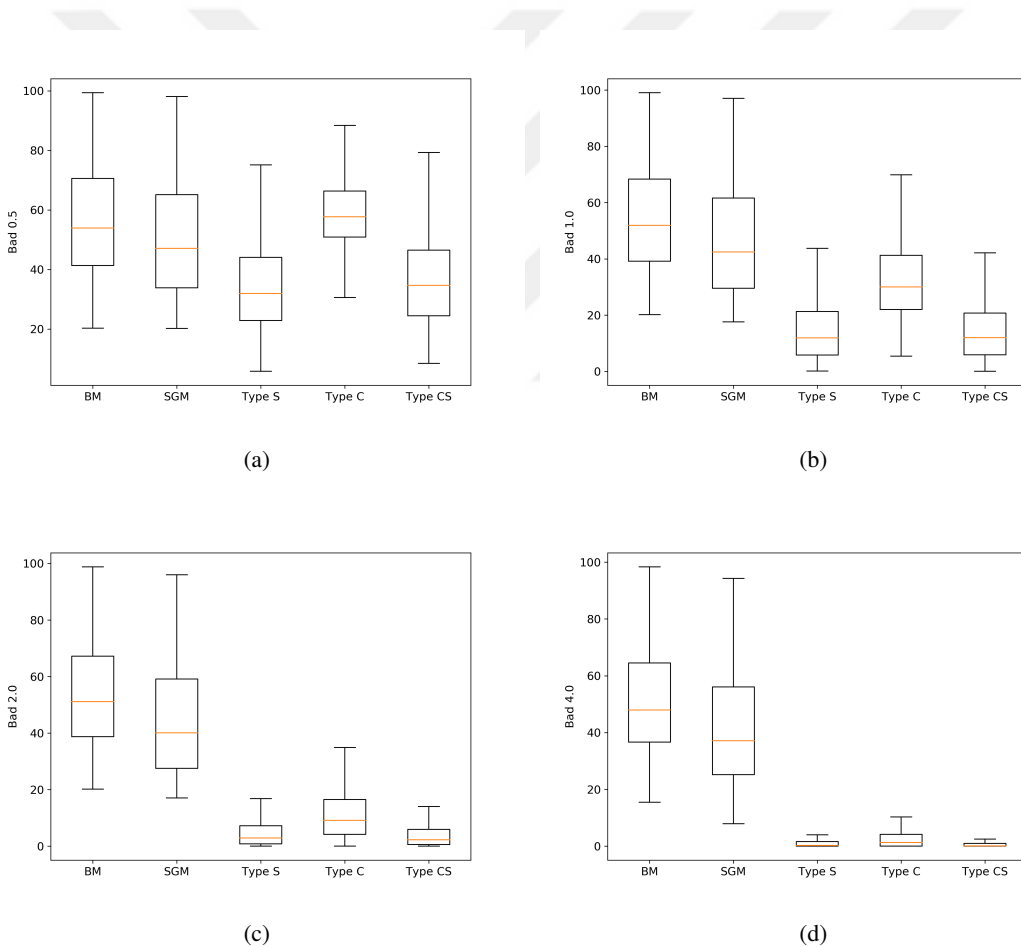


Figure 5.17: The box plots comparing the performance of the different methods. In this plot the lower value is better. **(a)** Bad 0.5 Metric. **(b)** Bad 1.0 Metric. **(c)** Bad 2.0 Metric. **(d)** Bad 4.0 Metric.

5.3.4 TMI Dataset Results

In their article, the creators of TMI dataset use root mean square (RMS) distance between corresponding points as the accuracy metric. We will use the same metric

since it is the output of the validation tool. By inputting the calculated point clouds into the validation tool, we will get a CSV file containing the statistics. We will be summarizing the statistics here.

As mentioned in Section 4.4.1, This dataset have been partitioned into sub-datasets like

1. Varying Distances.
2. Presence of smoke in view.
3. Presence of blood in view.
4. Varying angles.

Since our simulations are not focused on distance or angle, we will present their result later.

The evaluation result of the dataset, including all the partitions, is reported in Figure 5.18. There is a total number of 35 images in this category. This box plot shows that the performance of the Type C and Type S networks rivals the performance of Block Matching or Semi-Global Matching. There are no significant differences between their means. However, we are also interested in the consistency of the whole data, and the deviation of the error rate is much lower in Type S and Type C networks.

The performance of Type CS network is falling behind the rest of the method. We will explore the reason in Section 5.3.5. For now, it is sufficient to say that Type CS network, which had acceptable performance in the validation, was unable to generalize its knowledge.

The evaluation result of the dataset for all the images with smoke in them is reported in figure 5.19. There is a total number of 5 images in this category. As it is shown in this plot, there is a significant difference in the performance of the Type S and Type C network, comparing to the control methods. Even between two successful networks, Type C network is significantly outperforming the Type S network. Type CS network has the same issue which will be explored later.

The figure 5.20 compares the performance of all methods on the same surfaces with and without the smoke. It is expected that all the algorithms perform better without smoke than with smoke, which is validated by the result. We can observe that the performance of Type C network is better compared to all the other methods.

The evaluation result of the dataset for all the images with blood in them is reported in Figure 5.21. There is a total number of 2 images in this category. From the box plot, it appears that the Block Matching algorithm performs better than the rest, but it is little to none difference between the results (Type CS is an exception).

However, since the data points are too few ($n = 2$), this result is prone to statistical variation and mistakes accompanied by the datasets with too few data points. Nonetheless, we have included this result because it is helpful and necessary to acknowledge the problems related to any research, whether it is from the methods or the data.

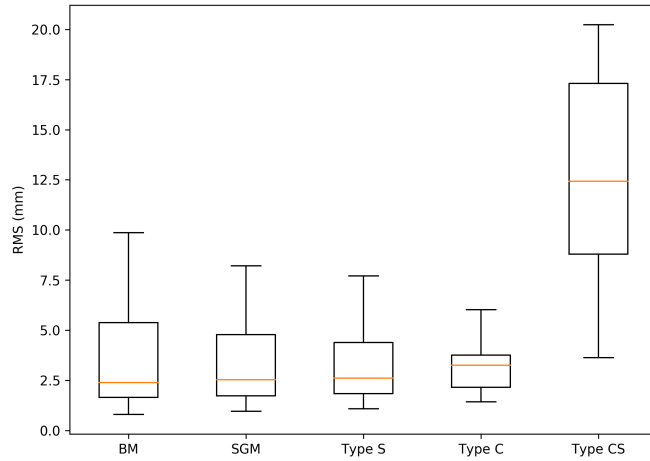


Figure 5.18: The RMS distance (mm) of all the images ($n = 35$) withing the TMI dataset with their ground truth.

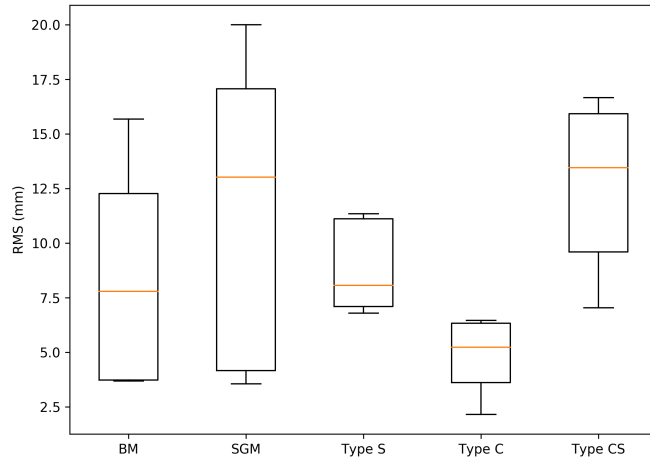


Figure 5.19: The RMS distance (mm) of all the images with smoke in view ($n = 5$), withing the TMI dataset with their ground truth.

The figure 5.22 compares the performance of all methods on the same surfaces with and without the blood. As it is said before, the results are inconclusive because of the limitation in the data.

Other than smoke and blood in the scene, the creators of the dataset included data with different angular views and different distances. Although we have not simulated such behaviors, in our training data, we will present the results.

Figure 5.23 shows a bar plot comparing the performances of different methods be-

tween a 0° and 30° views. In this image, we can observe that the performance of the Type C and Type S networks are better, as overall performance.

Additionally, Figure 5.24 shows a bar plot comparing the performance of different methods between images captured as $7cm$ and $4cm$. In this image, we can observe that the networks' performances are not consistent. The performance of Type C network may be affected by many factors, and since we are not controlling this parameter (distance), we cannot argue on the result or why one is drastically different than the other.

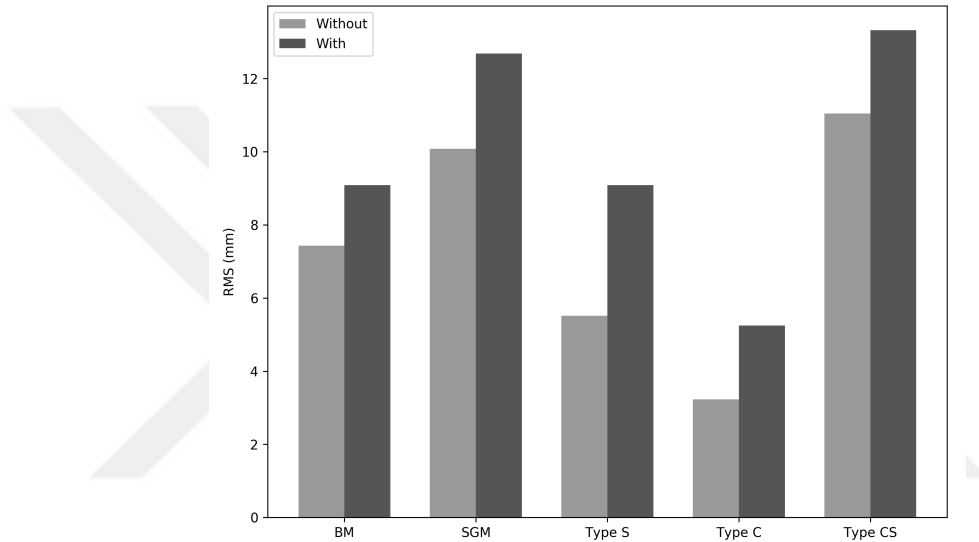


Figure 5.20: The RMS distance (mm) of all the images with smoke in view ($n = 5$), withing the TMI dataset with their ground truth.

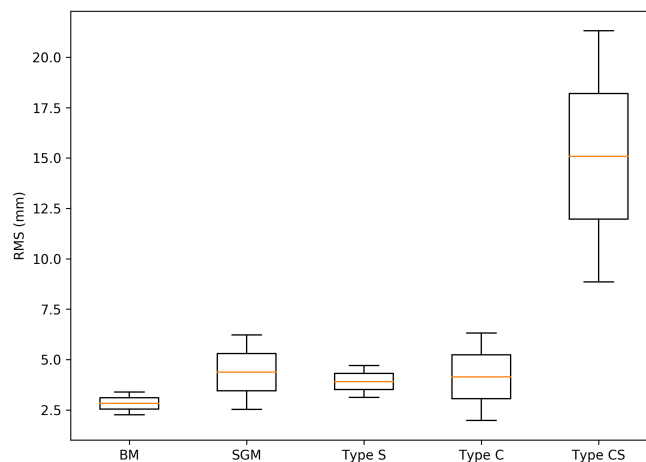


Figure 5.21: The RMS distance (mm) of all the images with blood in view ($n = 2$), withing the TMI dataset with their ground truth.

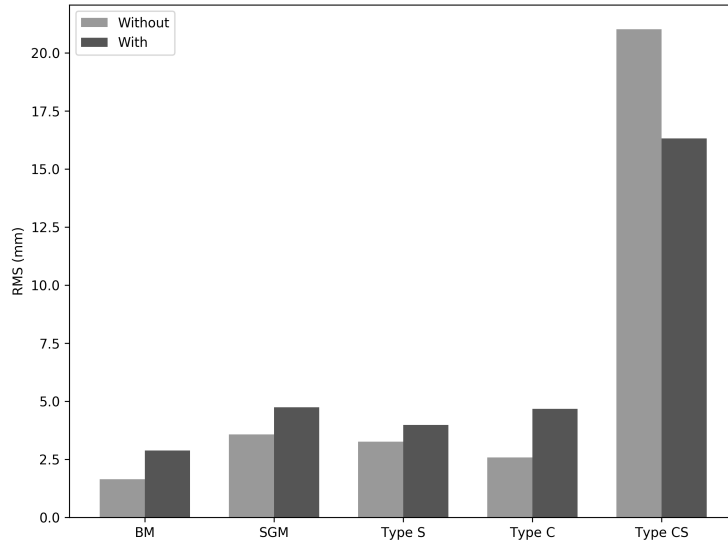


Figure 5.22: The RMS distance (mm) of all the images with blood in view ($n = 2$), withing the TMI dataset with their ground truth.

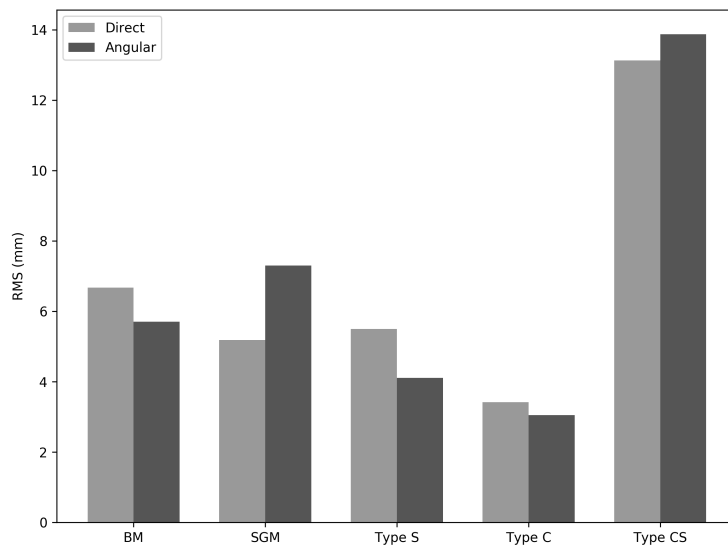


Figure 5.23: The RMS distance (mm) of all the images with direct and angular views ($n = 10$), withing the TMI dataset with their ground truth.

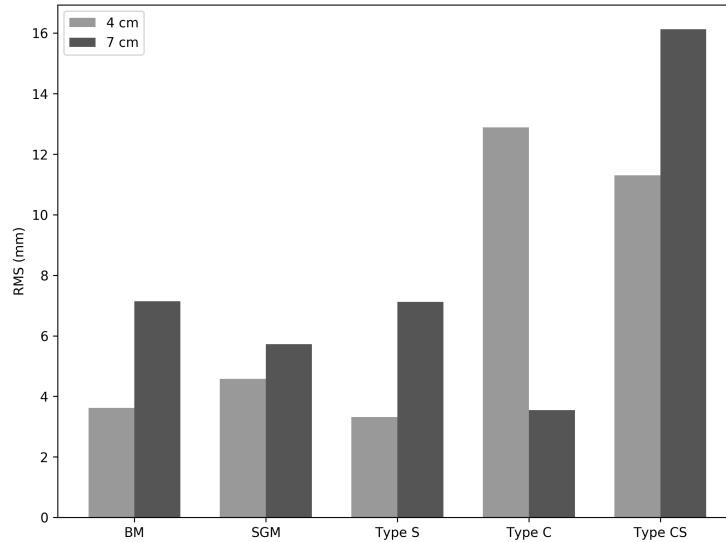


Figure 5.24: The RMS distance (mm) of all the images with different distances ($n = 10$), with their ground truth.

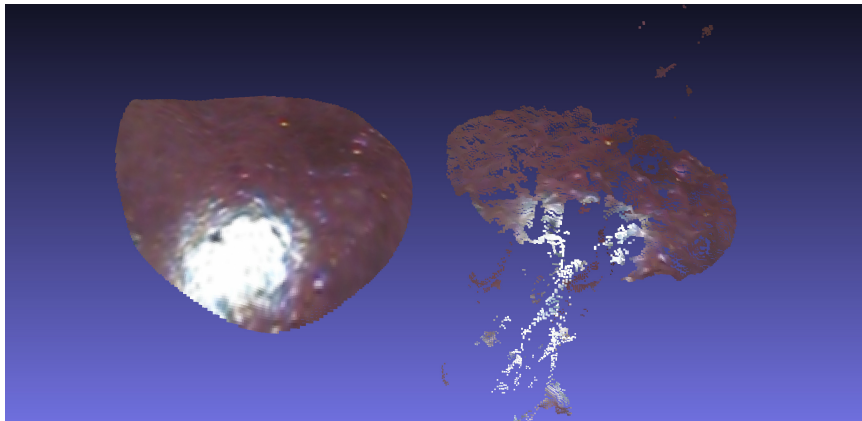
Visual Evaluation

Although the statistical result is very important in the evaluation of the methods, we believe an examination of the result for visual clues of performance, advantages, and shortcomings of the methods is also important. Using statistics to assign a numeric value of performance to a more complex output from a method and that may not be enough to evaluate the local and partial performance of each method in different regions of the image. The images below are comparing the Type C network and the Semi-Global Matching [36] results. These results would be in point cloud format. Figure 5.25 shows some examples.

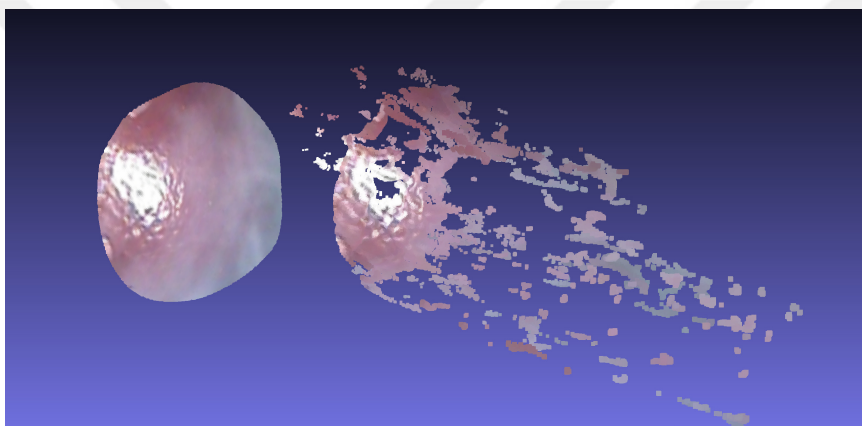
Figure 5.25a contains a well lit environment with reflection on the surface. The result of Type C network (left) is very smooth in comparison to the result of Semi-Global Matching (right). Moreover, the surface on which the reflection appears have been matched to a point much lower than the actual surface. We discussed this problem in section 4.3.

Figure 5.25b contains an example with considerable amount of smoke in between the camera and the surface. The Semi-Global Matching (right) has matched the smoke covered surface to a location above the actual surface. Also, this method matched the surface with reflection to a location lower than the actual surface. However, the Type C network (left) resulted in a consistence surface.

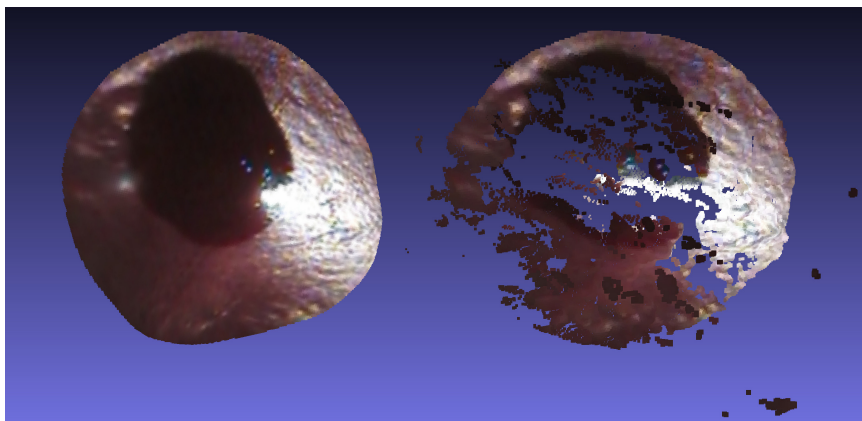
Figure 5.25c contains a surface with a patch of blood on it. This patch of blood will cover the features of the surface since it is featureless, conventional methods will have a hard time matching it. The Semi-Global Matching (right) has completely mismatched the surface covered with blood and the reflection along side it to a location beneath the actual surface. However, the Type C network (left) resulted in a consistent surface.



(a)



(b)



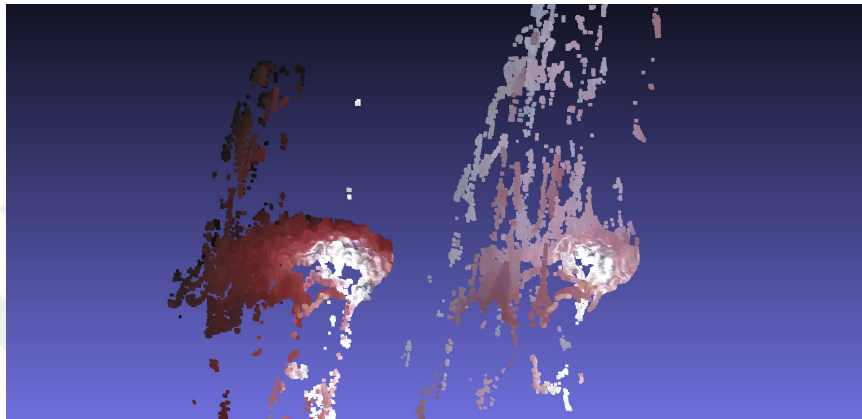
(c)

Figure 5.25: Comparing In-Vitro Reconstructions by Type C network (left) and Semi-Global Matching (right). **(a)** Specular Reflection on the surface. **(b)** Smoke covering the surface. **(c)** Blood on the surface.

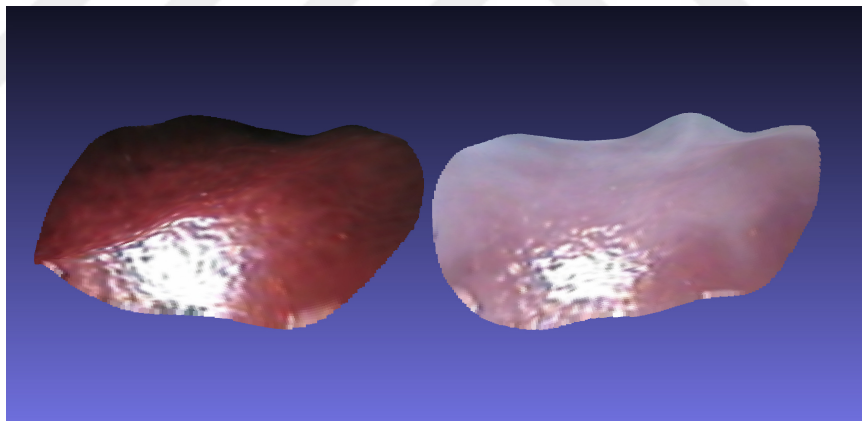
Since the TMI dataset contains data points of the same location with and without special conditions like presents of blood and smoke, we can also compare each algorithm's end results with and without those conditions.

For example Figure 5.26a compares the result of Semi-Global Matching algorithm with (right) and without (left) smoke in the view. As expected, the matching is severely affected on the smoke covered surface. Also, the locations with the reflection and the location with low illumination are mismatched.

However, as shown in figure 5.26b, the Type C network results in a consistency surface which occupies the same space in both cases - not considering small fluctuations.



(a)



(b)

Figure 5.26: Comparing In-Vitro Reconstructions of a target scene with smoke (right) and without smoke (left). **(a)** Reconstruction of Semi-Global Matching. **(b)** Reconstruction of Type C Network.

Viewing a point cloud and comparing them on a two-dimensional images is not ideal and Figures 5.25 and 5.26 might not convey the required information. Hence, The results from all reconstructions of the TMI dataset's data points (35 pairs of images) are available at <https://ms-thesis-result.aminzabardast.com>.

5.3.5 Type CS Network's Generalization Problem

In Chapter 5.3.4, we briefly mentioned that Type CS Network's performance is falling behind the rest of the networks, despite that it was performing well in the validation

phase. We argue that the reason behind this behavior is the higher capacity of the Type CS Network, alongside the simplicity of the synthesized dataset both in quantity and in variation.

The Type CS Network have eventually learned to memorize the examples given to it. Since the validation and training sets were not much different than each other, this issue was not detected during the validation.

Moreover, if the simulated dataset becomes more complex then the Type CS Network should be able to outperform the simpler networks, but in the current dataset, the simple Type S and Type C Networks are having better results. This is because their capacity is better matching the complexity of the dataset and they can generalize their knowledge base.

The possible improvements to the generated dataset are discussed in Section 6.3.

5.4 Summary

We created three different types of deep neural networks to test the validity of the dataset. Their architectures are based on the Auto-Encoder structures. This was because these structures can be designed to learn the whole process of the stereo matching without using other algorithms to complete the process.

The result of the networks is disparity maps, so a projection to three-dimensional space was required to make the data compatible with the TMI dataset validation tool. The evaluation was done in different stages. Initially, the dataset was evaluated as a whole, in which the performance of the networks was slightly better than the control algorithms. Type CS network was an exception, which will be addressed in the discussion section.

The TMI dataset has four different subsets. We evaluated the presence of the blood and smoke in the scene first since we included these variables in our simulations. In the case of the smoke, the performance of the networks was way better than the result of the control methods. But the result on the blood data was inconclusive. This can be traced back to the insignificant number of data points in the case of blood.

Additionally, there are two other subsets, one on different distances of the camera from the surface and the other on the different viewing angles. The overall performance on different angles was satisfactory in the case of the networks. However, the networks results varied on the distance subset.

One of the important results was that the results of the networks contain less variation in the error values, and the networks perform much better when there is not enough lighting in the scene, or the smoke has covered the surface of the organ.

CHAPTER 6

CONCLUSION

The main purpose of this research study is to reconstruct a three-dimensional representation of a surgical environment using Deep Neural Networks. The network will be using a pair of stereo images as input values to calculate the depth information. The core problem we are solving here is called the Stereo Correspondence problem, and the output from solving this problem is the depth information for each pixel on the camera images. This information is crucial to achieving Augmented Reality (AR) during surgical navigation.

The target surgical environment of this research is abdomen Minimally Invasive Surgery (MIS). However, working on the abdomen environment is difficult in comparison to other environments. Currently, the latest and most accurate methods that solve similar problems (Stereo Correspondence problem on other domains like Autonomously Driving Vehicles) are mostly Deep Neural Networks. However, Deep Neural Networks especially supervised learning methods, require a vast amount of data. Unfortunately, There are no public datasets available in the surgical domain that meets the requirements of a viable dataset to utilize.

To solve the dataset problem, we proposed to use in-silico simulations to create the dataset. In the past few years, the interest in using a synthesized dataset to train Deep Neural Networks is rising amount researcher. This approach is very novel, and there has not been much research around this subject. The goal is to create a dataset that a machine learning algorithm can experience and create a knowledge base from it, which can be used in a real situation.

We synthesized around 10.000 data points (stereo images with their ground truth) to use in the training process. These images are modeled after the data points in TMI dataset. This is because we are aiming to test the result on this dataset to evaluate if the networks were able to generalize their knowledge and use in real situations. We chose the preceding as our hypothesis.

After the data simulation, we attempted to create three different Deep Neural Networks based on Auto-Encoder architecture. Their purpose is to evaluate the learning process and answer the hypothesis questions. The Type S network is a simple Auto-Encoder structure. This network is trusted to experience the raw data, and autonomously create its knowledge base. The Type C network is utilizing a specialized layer, called correlational layer. The correlational layer is a specialized layer that calculates the correlation for small patches of the input images. This network is sup-

ported with this extra information, and the core structure is the same as the previous network (Auto-Encoder). The CS network is an ensemble of the Type C and Type S networks in the same order. The reason behind this structure was to evaluate the behavior of a considerably deeper network.

Additionally, we used two other methods as control methods to be able to compare the result of the networks to other established algorithms. These two methods are Block Matching and Semi-Global Matching [36].

6.1 Summary Of Results

Figure 5.12 and 5.13 shows Loss per Epoch plots of the three networks during training and evaluation.

We have used performance metrics suggested by [84] to evaluate the performance of the networks. Figure 5.15 is a box plot of the performance metrics.

It is clear that the networks are performing well on the validation dataset. Afterward, we tested our network's performances against the two control methods. Unfortunately, since the creators of the TMI dataset provide their validation tool for the dataset, we could not use our metrics. The metrics that this dataset uses are the root mean squared (RMS) distance between the correspondence points in the ground truth and the result. Figure 5.17 shows the comparison of the methods for all the data points.

Furthermore, we also evaluated the result of the methods of smoke and blood subsets separately. Their result are shown in figures 5.19 and 5.21.

6.2 Discussion

The overall decrease, visible in the figure 5.18, is the sign that the networks are successfully creating knowledge about the dataset. Among the networks, Type C has the most stable learning curve.

As it is shown in figure 5.19, the networks perform significantly better in case of smoke. The results in the case of blood are inconclusive since the data points were too few (2 data points).

Although establishing the realism of the synthesized dataset is not a part of this study, it should be mentioned that it is possible to do this by asking expert opinions. Creating a life like dataset is not the goal, neither it is necessary to spend resources achieving hyperrealism. As we see in this study, using our dataset can yield the required result. However, this should be discussed for the sake of completeness.

One can argue that using more and more realistic data can yield to better result. However, there is a certain trade off between spending the resources on hyperrealism and the expected improvements on the models being trained by this data.

In either case, we can evaluate the realism of our dataset by asking expert opinions through a rating procedure of the images within the dataset. Although, this process of rating is subjective, it is the most available scientific method at hand for evaluation of realism.

We should elaborate more on the issue of Type CS network failing to achieve an adequate generalization. We observed that Type CS Network fails to generalize its knowledge, while Type C and Type S networks are successful. This is the result of overfitting. The higher potential of the CS network has caused the network to memorize the data point of the training dataset. Hence, this network performs very well on validation but fails in the TMI dataset. A possible solution to this problem could be a larger and more varied dataset. However, larger quantities of data always improve the learning process, and it is advised to have as much data as possible.

Additionally, the specific dimensions of our Auto-encoder structure's bottleneck ($8 \times 8 \times 1024$) may also cause overfitting. The low spatial resolution (8×8) contributes to this issue by removing the finer detail of the simulated images. The dimensions of the bottleneck can be set up in a way to maximize saving of the finer details while avoiding the over or underfitting.

6.3 Future Studies

The positive results of this study and the existing research around the use of data simulation in Deep Learning approach to a practical problem show that there is potential in this research area. A possible future path to this study can be on improvement of the synthesized dataset. This can be achieved in multiple ways:

1. An improvement in the performance of the methods trained by the synthesized data can be achieved by synthesizing more realistic data. The recent advances in computer graphics, as well as the availability of powerful computational platforms, makes this approach more feasible. We argue that creating lifelike images are not necessary, and a compromise between realistic footage and easier synthesizing process can be achieved. Nonetheless, if the computational and creating process is not expensive, it is better to seek more realistic images.
2. The current version of the dataset was generated manually. This is the standard procedure in most of the synthesized dataset. However, This is time-consuming and prone to the bias of the creator. It is possible to create a procedural generation system to generate unique data points upon request. The algorithm would take a randomized hashed key and generates the environment which ultimately renders into stereo images and ground truth information. This process would be a quasi-random process, meaning that the process will be guided by some rules to generate the required surgical scene.

This procedural generation system will enable to use of adversarial approach to deep learning.

3. Understanding the theoretical framework of synthetic datasets is important. This knowledge can guide the generation process and help to synthesize a

dataset that is fit to replace a real dataset for a real problem.

Unfortunately, there are no theoretical studies on this subject. However, as it was mentioned before, more researchers are getting interested in the applications of synthesized datasets, and there is a growing amount of research around this subject.

For example, the ability to quantify the realness of a dataset will be very useful. Ideally, if we can compare two sets of data on how similar they are holistically, then we can use this metric in the generation process. Then, some real data can be used as seeds to generate more synthesized data.

Additionally, understanding the effects of different dataset characteristics will be useful. This knowledge can help to fine-tune the data generation process.

For example, by accumulating this knowledge, we can create a dataset that is varied enough to be a fair representation of its statistical population, and it can also help by normalizing the distribution of the data points to avoid unbalanced datasets.

4. We have already mentioned that the Stereo Correspondence problem fit perfectly with a Deep Learning solution. Creating better architectures that can understand and process the stereo input better and can yield a better result with the data at hand can be a path to continue this research.

Experimenting with the depth (channels) and spatial resolution of the bottleneck can help in achieving better results while using the same training dataset.

5. A human's depth perception is a combination of visual clues and deduction of the information. For example, we use our binocular vision to understand the depth information from the disparities between the images our left and right eyes receive. This is an approach like many of the methods discussed in Chapter 2.

However, when a three-dimensional space is mapped into several two-dimensional representations, a huge amount of information will be lost. So going back to a three-dimensional representation is not easy and needs processing for reconstruction. Humans use the objects in their view and their knowledge about the shapes and the orientation of objects to deduce the depth information and reconstruct a mental image of the scene.

We can follow the same strategy by using a hybrid system containing a simple stereo matching method and a Deep Neural Network component. The idea would be to use the stereo matching method to feed seed point to the neural network and then trust the neural network to reconstruct a dense depth map by deducing the depth information from the images and the seeds provided as it input.

REFERENCES

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] H. A. Alhaja, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126(9):961–972, 2018.
- [3] P. Aschwanden, W. Guggenbuhl, et al. Experimental results from a comparative study on correlation-type registration algorithms. *Robust computer vision*, pages 268–289, 1992.
- [4] R. T. Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.
- [5] F. Bagante, G. Spolverato, S. M. Strasberg, F. Gani, V. Thompson, B. L. Hall, D. J. Bentrem, H. A. Pitt, and T. M. Pawlik. Minimally invasive vs. open hepatectomy: a comparative analysis of the national surgical quality improvement program database. *Journal of gastrointestinal surgery*, 20(9):1608–1617, 2016.
- [6] F. L. Bauer. Computational graphs and rounding error. *SIAM Journal on Numerical Analysis*, 11(1):87–96, 1974.
- [7] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.
- [8] P. N. Belhumeur. A bayesian approach to binocular stereopsis. *International Journal of Computer Vision*, 19(3):237–260, 1996.
- [9] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [10] R. Benosman and J. Devars. Panoramic stereo vision sensor. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, volume 1, pages 767–769. IEEE, 1998.
- [11] S. Bernhardt, S. A. Nicolau, L. Soler, and C. Doignon. The status of augmented reality in laparoscopic surgery as of 2016, 2017.

- [12] A. Bhatti. *Current advancements in stereo vision*. InTech, 2012.
- [13] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):1124–1137, 2004.
- [14] L. Breiman. *Classification and regression trees*. Routledge, 2017.
- [15] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):993–1008, 2003.
- [16] J.-R. Chang and Y.-S. Chen. Pyramid Stereo Matching Network. *arXiv preprint arXiv:1803.08669*, 2018.
- [17] X. Cheng, P. Wang, and R. Yang. Learning depth with convolutional spatial propagation network. *arXiv preprint arXiv:1810.02695*, 2018.
- [18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.
- [19] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer vision and image understanding*, 63(3):542–567, 1996.
- [20] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision*, 37(2):175–185, 2000.
- [21] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.
- [22] A. Dosovitskiy, P. Fischery, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 2758–2766, 2015.
- [23] X. Du, M. El-Khamy, and J. Lee. Amnet: Deep atrous multiscale stereo disparity estimation networks. *arXiv preprint arXiv:1904.09099*, 2019.
- [24] C. R. Dyer. Volumetric scene reconstruction from multiple views. In *Foundations of image understanding*, pages 469–489. Springer, 2001.
- [25] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.
- [26] A. Gaidon, A. Lopez, and F. Perronnin. The reasonable effectiveness of synthetic visual data. *International Journal of Computer Vision*, 126(9):899–901, 2018.
- [27] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013.

- [28] S. Giannarou, M. Visentini-Scarzanella, and G.-Z. Yang. Probabilistic tracking of affine-invariant anisotropic regions. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):130–143, 2012.
- [29] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [30] N. Haouchine, J. Dequidt, I. Peterlik, E. Kerrien, M.-O. Berger, and S. Cotin. Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery. In *2013 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 199–208. IEEE, 2013.
- [31] A. G. Harrell and B. T. Heniford. Minimally invasive abdominal surgery: lux et veritas past, present, and future. *The American journal of surgery*, 190(2):239–243, 2005.
- [32] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [33] R. I. Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, 1999.
- [34] H. Hattori, N. Lee, V. N. Boddeti, F. Beainy, K. M. Kitani, and T. Kanade. Synthesizing a scene-specific pedestrian detector and pose estimator for static video surveillance. *International Journal of Computer Vision*, 126(9):1027–1044, 2018.
- [35] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
- [36] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.
- [37] M. C. Hornbrook, R. Goshen, E. Choman, M. O’Keeffe-Rosetti, Y. Kinar, E. G. Liles, and K. C. Rust. Early colorectal cancer detected by machine learning model using gender, age, and complete blood count data. *Digestive diseases and sciences*, 62(10):2719–2727, 2017.
- [38] K. Huh, J. Park, J. Hwang, and D. Hong. A stereo vision-based obstacle detection system in vehicles. *Optics and Lasers in Engineering*, 46(2):168–178, 2008.
- [39] C. Jiang, S. Qi, Y. Zhu, S. Huang, J. Lin, L.-F. Yu, D. Terzopoulos, and S.-C. Zhu. Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision*, 126(9):920–941, 2018.
- [40] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1088–1095. IEEE, 1991.
- [41] W. Kazmi, S. Foix, G. Alenyà, and H. J. Andersen. Indoor and outdoor depth imaging of leaves with time-of-flight and stereo vision sensors: Analysis and comparison. *ISPRS Journal of Photogrammetry and Remote Sensing*, 88:128–146, 2014.

- [42] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 573–590, 2018.
- [43] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.
- [44] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [45] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. Technical report, Cornell University, 2001.
- [46] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.
- [47] C. Lee, H. Song, B. Choi, and Y. S. Ho. 3D scene capturing using stereoscopic cameras and a time-of-flight camera. *IEEE Transactions on Consumer Electronics*, 57(3):1370–1376, 2011.
- [48] D. B. Lenat and R. V. Guha. Building large knowledge-based systems; representation and inference in the cyc project. 1989.
- [49] M. Lerotic, A. J. Chung, J. Clark, S. Valibeik, and G.-Z. Yang. Dynamic view expansion for enhanced navigation in natural orifice transluminal endoscopic surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 467–475. Springer, 2008.
- [50] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 125–131. IEEE, 1999.
- [51] G. Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.
- [52] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [53] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.
- [54] X. Luo, A. J. McLeod, S. E. Pautler, C. M. Schlachta, and T. M. Peters. Vision-based surgical field defogging. *IEEE transactions on medical imaging*, 36(10):2021–2030, 2017.
- [55] L. Maier-Hein, A. Groch, A. Bartoli, S. Bodenstedt, G. Boissonnat, P. L. Chang, N. T. Clancy, D. S. Elson, S. Haase, E. Heim, J. Hornegger, P. Janin, H. Kenngott, T. Kilgus, B. Muller-Stich, D. Oladokun, S. Rohl, T. R. Dos Santos, H. P. Schlemmer, A. Seitel, S. Speidel, M. Wagner, and D. Stoyanov. Comparative validation of single-shot optical techniques for laparoscopic 3-d

- surface reconstruction. *IEEE Transactions on Medical Imaging*, 33(10):1913–1930, 2014.
- [56] J. Mallon and P. F. Whelan. Projective rectification from the fundamental matrix. *Image and Vision Computing*, 23(7):643–650, 2005.
- [57] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, pages 3061–3070, 2015.
- [58] M. Menze, C. Heipke, and A. Geiger. Joint 3D Estimation of Vehicles and Scene Flow. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W5:427–434, 2015.
- [59] T. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.
- [60] P. Mountney, D. Stoyanov, and G.-Z. Yang. Three-dimensional tissue deformation recovery and tracking. *IEEE Signal Processing Magazine*, 27(4):14–24, 2010.
- [61] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, pages 1–18, 2018.
- [62] Y. Munz, K. Moorthy, A. Dosis, J. D. Hernandez, S. Bann, F. Bello, S. Martin, A. Darzi, and T. Rockall. The benefits of stereoscopic vision in robotic-assisted performance on bench models. *Surgical Endoscopy and Other Interventional Techniques*, 18(4):611–616, 2004.
- [63] D. Murray and J. Little. Using Real-Time Stereo Vision for Mobile Robot Navigation. *Autonomous Robots*, 8:161–171, 2000.
- [64] K. W. Nam, J. Park, I. Y. Kim, and K. G. Kim. Application of stereo-imaging technology to medical field, 2012.
- [65] S. K. Nayar and M. Gupta. Diffuse structured light. In *2012 IEEE International Conference on Computational Photography, ICCP 2012*, 2012.
- [66] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.
- [67] Y. Ohta and T. Kanade. Stereo by intra-and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2):139–154, 1985.
- [68] A. O. Ozturk. 3D Face Reconstruction Using Stereo Images And Structured Light. Master’s thesis, Middle East Technical University, 2007.
- [69] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan. Cascade Residual Learning: A Two-stage Convolutional Neural Network for Stereo Matching. In *International Conf. on Computer Vision-Workshop on Geometry Meets Deep Learning (ICCVW 2017)*, volume 3, 2017.

- [70] H. Park. An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing*, 43(2):154–164, 2013.
- [71] Y. Peng, A. Rysanek, Z. Nagy, and A. Schlüter. Using machine learning techniques for occupancy-prediction-based cooling control in office buildings. *Applied energy*, 211:1343–1358, 2018.
- [72] V. Penza, A. S. Ciullo, S. Moccia, L. S. Mattos, and E. De Momi. Endoabs dataset: Endoscopic abdominal stereo image dataset for benchmarking 3d stereo reconstruction algorithms. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 14(5):e1926, 2018.
- [73] P. Pratt, D. Stoyanov, M. Visentini-Scarzanella, and G.-Z. Yang. Dynamic guidance for robotic surgery using image-constrained biomechanical models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 77–85. Springer, 2010.
- [74] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- [75] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
- [76] A. H. Renear, S. Sacchi, and K. M. Wickett. Definitions of dataset in the scientific and technical literature. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem-Volume 47*, page 81. American Society for Information Science, 2010.
- [77] I. Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.
- [78] G. Rogez and C. Schmid. Image-based synthesis for deep 3d human pose estimation. *International Journal of Computer Vision*, 126(9):993–1008, 2018.
- [79] S. Röhl, S. Bodenstedt, S. Suwelack, R. Dillmann, S. Speidel, H. Kenngott, and B. P. Müller-Stich. Dense GPU-enhanced surface reconstruction from stereo endoscopic images for intraoperative registration. *Medical Physics*, 39(3):1632–1645, 2012.
- [80] A. Rosenfeld. Picture processing by computer. *ACM Computing Surveys (CSUR)*, 1(3):147–176, 1969.
- [81] A. Rosenfeld and A. C. Kak. Digital picture processing. *Computer Science and Applied Mathematics*, 1:116–204, 1982.
- [82] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [83] C. Sakaridis, D. Dai, and L. Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, pages 1–20, 2018.

- [84] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001*, pages 131–140, 2001.
- [85] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.
- [86] Y.-H. Seo, J.-S. Yoo, and D.-W. Kim. A new parallel hardware architecture for high-performance stereo matching calculation. *Integration, the VLSI Journal*, 51:81–91, 2015.
- [87] L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, 2001.
- [88] Z. Song. *Handbook of 3D machine vision*. 2013.
- [89] D. Stoyanov, G. P. Mylonas, F. Deligianni, A. Darzi, and G. Z. Yang. Soft-tissue motion tracking and structure estimation for robotic assisted mis procedures. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 139–146. Springer, 2005.
- [90] D. Stoyanov, M. V. Scarzanella, P. Pratt, and G.-Z. Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 275–282. Springer, 2010.
- [91] J. Sun, N.-N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):787–800, 2003.
- [92] C. Tomasi and R. Manduchi. Stereo without search. In *European Conference on Computer Vision*, pages 452–465. Springer, 1996.
- [93] C. Tomasi and R. Manduchi. Stereo matching as a nearest-neighbor problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):333–340, 1998.
- [94] V. Venkateswar and R. Chellappa. Hierarchical stereo and motion correspondence using feature groupings. *International Journal of Computer Vision*, 15(3):245–269, 1995.
- [95] M. Ye, S. Giannarou, A. Meining, and G.-Z. Yang. Online tracking and retargeting with applications to optical biopsy in gastrointestinal endoscopic examinations. *Medical image analysis*, 30:144–157, 2016.
- [96] M. Ye, E. Johns, A. Handa, L. Zhang, P. Pratt, and G.-Z. Yang. Self-supervised siamese learning on stereo image pairs for depth estimation in robotic surgery. *arXiv preprint arXiv:1705.08260*, 2017.
- [97] Z. Yin, T. Darrell, and F. Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6044–6053, 2019.

- [98] J. Žbontar and Y. Le Cun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 1592–1599, 2015.
- [99] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1592–1599, 2015.
- [100] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019.
- [101] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.
- [102] Y.-T. Zhou and R. Chellappa. Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks*, volume 1998, pages 71–78, 1988.