

3D Endoscopy: Depth Prediction for Endoscopic Scenes

A Thesis

submitted by

D TONY FREDRICK

*in partial fulfilment of the requirements
for the award of the degree of*

BACHELOR AND MASTER OF TECHNOLOGY



**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.**

June 17, 2022

THESIS CERTIFICATE

This is to certify that the thesis titled **3D Endoscopy: Depth Prediction for Endoscopic Scenes**, submitted by **D Tony Fredrick**, to the Indian Institute of Technology, Madras, for the award of the degree of **Dual Degree (Bachelor of Technology + Master of Technology)**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Kaushik Mitra
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600036

Place: Chennai
Date: June 17, 2022

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to my guide, Dr Kaushik Mitra, for his constant support, encouragement and guidance throughout my Dual Degree Project. He has always been a pillar of support and has been a great mentor to me. I am grateful to him for providing this excellent opportunity to work on an exciting project. I thank him for providing a nurturing atmosphere and unique opportunities for growth in my professional journey.

I express my gratitude to Salman Siddique Khan, a PhD researcher at the Computational Imaging Lab, IIT Madras. He is always brimming with new ideas, and it was through his insights and mentorship that I have been able to complete this project. My weekly meetings with him helped me learn how to solve research problems. I have learnt a lot from his knowledge, patience, perseverance and approach. Working with him has been an immense pleasure. I would also like to thank Balachander S for his help with designing and building the lab setup for photometric stereo data collection.

I would like to thank my institute, IIT Madras, for providing me with a holistic learning environment for personal and professional growth. I thank my friends, Dhruvjyoti Bagadthey, Sarah, Manogna Param Koolath, Sanjana S Prabhu and Suhas Morisetty, some of whom were also my labmates, for their support, encouragement and friendship that were instrumental in making my time at IIT Madras a memorable experience to cherish forever.

I express my gratitude to my beloved parents for being constant pillars of support in my life. Lastly, but most importantly, I thank the Almighty for showering his divine blessings on me.

ABSTRACT

KEYWORDS: Structure-from-Motion; 3D Endoscopy; Photometric Stereo; Un-supervised Learning; Monocular depth estimation; Photometric Stereo Endoscopic dataset

Endoscopic images provide information about the internal organs, such as those of the gastrointestinal tract. When it comes to detecting lesions and polyps, the topographical variation of the lesion as compared to its surroundings is more pronounced than the colour variation. Therefore, obtaining the three-dimensional information like pixel-wise depth and surface normals of endoscopic scenes can improve the detection and classification of lesions as cancerous or benign and also help in minimally invasive surgery. This project explores approaches based on Structure-from-Motion (SfM) and Photometric stereo for obtaining pixel-wise depth maps. We have used both traditional feature extraction and matching techniques and learning-based techniques. An end-to-end unsupervised learning method for joint estimation of depth and pose is also demonstrated. Following this, conventional and near-field photometric stereo methods have been tested. A major contribution of this project is the creation of a synthetic dataset using the Unity3D environment. It is a dataset of endoscopic scenes captured using rendered organs with variable illumination conditions. The details of the data generation pipeline are elucidated in this thesis. To our knowledge, this is the first large-scale photometric stereo dataset for endoscopic images. This dataset is used to train a fast, lightweight, near-field photometric stereo model to predict the surface normal maps and depth maps of endoscopic images. A detailed explanation of the lab setup that will be used for capturing photometric stereo images of ex-vivo tissues is also provided in this thesis.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
ABSTRACT	iii
LIST OF FIGURES	xi
1 Introduction	1
2 Correspondence-based methods: Structure from Motion	5
2.1 Overview of SfM	5
2.2 Contributions of COLMAP	6
2.3 Experiments using COLMAP	7
2.4 SuperPoint: Self-Supervised Interest Point Detection and Description	8
2.5 SuperGlue: Learning Feature Matching With Graph Neural Networks	9
2.6 Experiments using SuperPoint features and SuperGlue matching algorithm	10
3 Unsupervised Learning Approach	13
3.1 Introduction	13
3.2 Self-supervised Monocular Depth Estimation	14
3.3 Endo-SfMLearner	14
3.3.1 Working of the network:	15
3.3.2 Training objective and Loss function	16
3.3.3 Endo-SfMLearner spatial attention block (ESAB)	17
3.4 EndoSLAM dataset	17
3.5 Experiments	17
3.5.1 Depth Estimation:	17
3.5.2 Pose Estimation:	21
3.5.3 Error metrics	24
3.5.4 Summary	25

4	Photometric Stereo Endoscopy	27
4.1	Introduction	27
4.2	Limitations of Conventional Photometric Stereo	28
4.3	Calibration Method	28
4.3.1	Theory	29
4.3.2	Lab Setup for Calibration and Data Collection	29
4.4	Conventional Photometric Stereo	31
4.4.1	Theory	31
4.4.2	Experiments	31
5	Rendering a Photometric Stereo dataset for Endoscopic Images	37
5.1	Introduction	37
5.2	Data Simulation	37
5.2.1	Rendered organs	37
5.2.2	Camera and Lights Setup	38
5.3	Data Collection	39
5.3.1	Surface Normal Preprocessing	40
5.3.2	Getting Absolute Depth	41
5.4	Examples	41
6	Deep Learning for Photometric Stereo	45
6.1	Introduction	45
6.2	Overview	45
6.3	Experiments and Results	46
7	Conclusion and Future work	51
8	Appendix: Guide for Data Collection using Unity	55
8.1	Installation	55
8.2	Creating a new project and downloading the recorder:	56
8.3	Importing the organ models:	57
8.4	Camera Parameters	62
8.5	Data Collection using Unity	63
8.6	Steps for Data Collection	65

8.7	Text File data	67
8.8	Calibration setup	68

LIST OF FIGURES

1.1	Obtaining pixel-wise depth from a colour image	2
1.2	Improved visibility of lesions on using topographical information . .	2
1.3	Detection of metastatic melanoma	2
2.1	Incremental SfM pipeline (COLMAP)	6
2.2	Depth Estimation Pipeline	7
2.3	Depth map reconstructed by COLMAP	8
2.4	SuperPoint self-supervised training method	8
2.5	SuperPoint network architecture	9
2.6	SuperGlue network architecture	10
2.7	Using SuperPoint features and nearest-neighbour matching (endoscopic images)	11
2.8	Using SuperPoint features and SuperGlue for matching (endoscopic images)	11
2.9	Using SuperPoint features and nearest-neighbour matching	12
2.10	Using SuperPoint features and SuperGlue for matching	12
3.1	Pixel-wise depth predicted by Monodepth2 network	14
3.2	Working of the Endo-SfMLearner network	15
3.3	Endo-SfMLearner network architecture	16
3.4	EndoSLAM Depth Predictions for model trained on real data	18
3.5	EndoSLAM Depth Predictions for model trained on real data	19
3.6	EndoSLAM Depth Predictions for model trained on simulated data .	20
3.7	EndoSLAM Depth Predictions for model trained on simulated data .	21
3.8	Training and Validation Loss vs Epoch for Endo-SfMLearner	21
3.9	Estimated trajectory and ground-truth trajectory for first 150 frames of the Small Intestine dataset	23
3.10	Estimated trajectory and ground-truth trajectory for the Small Intestine dataset	24
4.1	Conventional Photometric Stereo Example	27

4.2	Light Direction Calibration	28
4.3	Lab Setup for Calibration and Data Capture	30
4.4	Ring of 24 LEDs around the camera	30
4.5	3 out of 24 calibration images of the specular ball	32
4.6	Photometric Stereo results on using all 24 images	32
4.7	Photometric Stereo results on using only 3 images	32
4.8	3 out of 24 images of the rabbit (Lambertian surface)	33
4.9	Photometric Stereo results on using all 24 images	33
4.10	Photometric Stereo results on using only 3 images	33
4.11	Photometric stereo images of a specular ball (simulated)	34
4.12	Photometric Stereo results for the rendered specular ball	34
4.13	Photometric stereo images of the colon (simulated)	35
4.14	Photometric Stereo results for the rendered colon	35
5.1	Lab setup using Raspberry Pi camera and LEDs	38
5.2	Diagrammatic representation of the Camera + LEDs model	39
5.3	Images of an endoscopic scene with 4 light sources	42
5.4	Surface Normal and Depth maps	42
5.5	Images of an endoscopic scene with 4 light sources	43
5.6	Surface Normal and Depth maps	43
5.7	Images of an endoscopic scene with 4 light sources	44
5.8	Surface Normal and Depth maps	44
6.1	3 out of 24 images of the rabbit (Lambertian surface)	46
6.2	Predictions using the pretrained model (Fast Near-Field Photometric Stereo)	46
6.3	Images of the sphere (rendered in Unity) under different illumination conditions	47
6.4	Predictions using the pretrained model (Fast Near-Field Photometric Stereo)	47
6.5	Images of the endoscopic scene (rendered in Unity) under different illumination conditions	47
6.6	Predictions using the pretrained model (Fast Near-Field Photometric Stereo)	48
6.7	Images of an endoscopic scene (rendered in Unity)	48

6.8	Predictions using the model trained on our rendered dataset	49
6.9	Images of another endoscopic scene (rendered in Unity)	49
6.10	Predictions using the model trained on our rendered dataset	50
6.11	Training and Validation curves indicating overfitting	50
8.1	Diagrammatic representation of the Camera + LEDs model	64

CHAPTER 1

Introduction

Cancer of the gastrointestinal (GI) tract affect more than 28 million people every year and this type of cancer represents about 26% of the worldwide cancer incidence and 35% of all deaths related to cancer [1]. The most effective and simple way to screen such cancers is by direct visual inspection (DVI). Many lesions that occur on the gastrointestinal tract that are clinically significant have very unique surface topographies. This topographical contract is even more pronounced than the variation in the colouration of these lesions as compared to their surroundings [2]. Such lesions, although difficult to diagnose using the conventional colour images, can protrude significantly into the surrounding tissue.

For analysis using conventional endoscopy, the colour variation of the tissues is the main source of contrast. In fact, the lighting strategies of an endoscope camera minimize the topographical contract. Conventional methods that rely only on colour variations are estimated to miss one out of every four lesions [3]. When the topographical features are pronounced, modifying the imaging hardware so that we can obtain both conventional colour images and the topography of the surface can greatly improve the detection and accuracy of classification of the lesions as cancerous or benign.

In addition to cancer diagnosis, the recovery of the three-dimensional structure of the organs and tissues viewed in endoscopic images is a very important computer vision task in minimally invasive surgery. A general solution to this problem has several applications such as enhanced intra-operative surgical guidance, novel-view synthesis, depth perception, 3D motion estimation and improving pre-operative/intra-operative data registration [4].

With these goals in mind, this project aims to produce three-dimensional information of the scenes captured by an endoscopic camera. This is achieved by producing the pixel-wise depth map for the endoscopic images of internal organs, which provides information about the internal structure and can then be used to detect lesions and tumours. The methods that have been implemented can be divided into two broad topics:

Structure from Motion (SFM) and Photometric Stereo. Structure-from-Motion can be used to produce a sparse point cloud; from which dense depth information can be obtained by stereo methods like plane sweep stereo. Photometric stereo can be used to produce dense depth and surface normal maps.

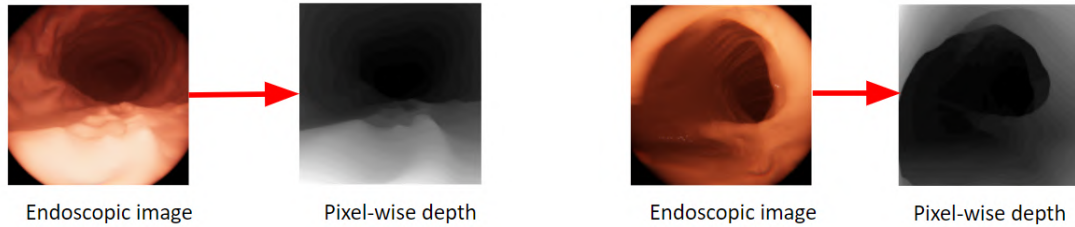


Figure 1.1: Obtaining pixel-wise depth from a colour image

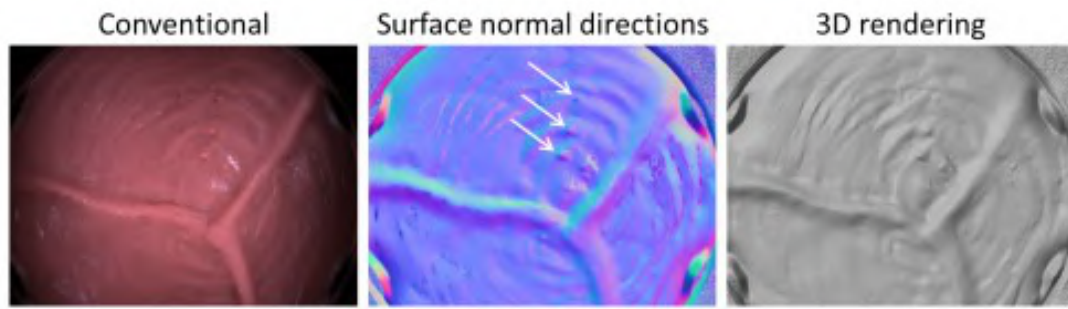


Figure 1.2: Improved visibility of lesions on using topographical information

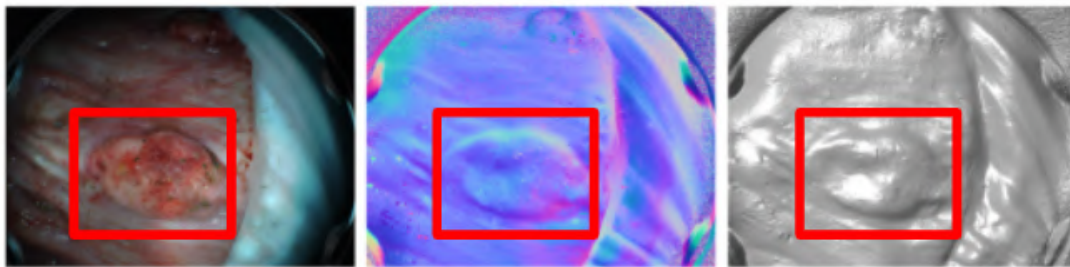


Figure 1.3: Detection of metastatic melanoma

Figure 1.1 shows the aim of the project, which is to produce pixel-wise depth maps. In Figure 1.2, the lesions which are not distinctly visible in the colour image, can be seen clearly after obtaining the 3D model from the surface normals. Figure 1.3 shows the detection of a metastatic melanoma which is once again very distinctly visible in the surface normal map and 3D rendered model.

The first method that I tried was COLMAP which is an improved version of the conventional structure from motion approach. This was followed by learning-based

methods for feature detection and matching, followed by dense depth map reconstruction. An end-to-end deep learning-based approach to jointly predict depth and relative pose was also used. Due to unavailability of ground truth depth for endoscopic images, I used an unsupervised training approach.

The second approach is based on Photometric Stereo. Both conventional and learning-based methods have been tested. Since there are no existing labelled photometric stereo datasets for endoscopic images, such a dataset has been rendered using Unity. A detailed description of the rendering method can be found in the appendix. Using this dataset, a light-weight, near-field photometric stereo deep learning model has been trained and the results have been shown in this thesis. The camera and lights setup used in the simulation mimic a prototype endoscope developed in our lab, (i.e. the Computational Imaging Laboratory, IIT Madras) the details of which are also explained in this thesis.

CHAPTER 2

Correspondence-based methods: Structure from Motion

2.1 Overview of SfM

Structure from Motion is the method of reconstructing the 3D structure of a scene from images of the scene taken from multiple viewpoints. The method predicts the camera poses and the sparse point cloud of feature points. The first step is correspondence matching between the input images in which the projections of the same points in the overlapping parts of images are identified. For each image I_i , SfM detects a feature set of local features at location x_j and, represented by the feature descriptor f_j . Feature descriptors could be of multiple types, in this thesis, SIFT feature descriptors and SuperPoint [5] feature descriptors have been used. SuperPoint feature descriptors involve a learning-based feature extraction approach.

Next, the method identifies images that view the same portion of the scene using the features detected in the feature description step as an appearance descriptor of the images. The feature descriptors can be compared using various conventional approaches like nearest neighbor matching or by using a learning-based method known as SuperGlue [6]. The expected output is a set of potentially overlapping image pairs and their associated feature correspondences. The matches between image pairs are geometrically verified by trying to estimate a transformation that maps features between images using projective geometry. If a valid transformation maps a sufficient number of features between the images, then they are said to be geometrically verified. Due to the presence of outliers in the feature descriptors, estimation techniques robust to outliers, such as RANSAC [7] are used.

The next step is the 3D reconstruction. A suitable pair is selected for the initial two-view reconstruction. New images are added to the existing model by solving the Perspective-n-Point problem using feature correspondences to triangulated points in already registered images (correspondences between 2D and 3D features). Every new image that is added to the model either observes already existing 3D points or increases

the scene coverage by adding more 3D points by triangulation. This crucial step increases the stability of the model through redundancy and it also gives more 2D-3D correspondences so that more images can be added into the model.

The uncertainties in the camera poses propagate to the triangulated points and the uncertainties in the triangulated points propagate to the camera poses. Therefore, there is a final bundle adjustment step which is a non-linear refinement of the camera poses and the triangulated point positions that minimizes the reprojection error.

2.2 Contributions of COLMAP

COLMAP [8] is a general-purpose Structure-from-Motion pipeline that improves over the conventional SfM method in several ways, as follows:

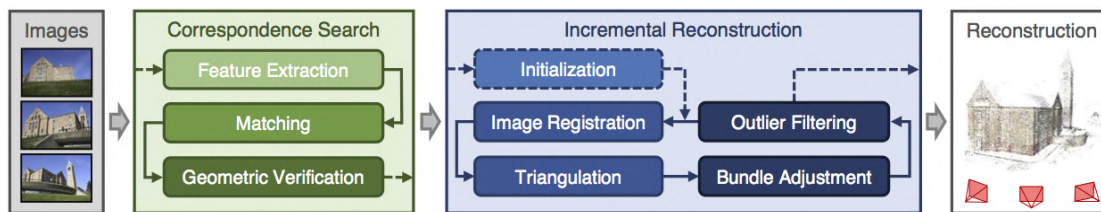


Figure 2.1: Incremental SfM pipeline (COLMAP)

- **Next Best View Selection:** At each step of the reconstruction, choosing the best view is important as it determines the accuracy of the rest of the reconstruction process. For each of the candidate views, COLMAP keep track of the feature points in the image that correspond to the already triangulated 3D points. The view from which the highest number of triangulated points are visible and in which these points are more uniformly distributed is chosen as the “next best view”.
- **Robust and Efficient Triangulation:** COLMAP leverages transitivity and establishes correspondences between images with larger baselines. This improves image registration and subsequent triangulation accuracy.
- **Bundle Adjustment:** Local bundle adjustment is performed on the set of the most connected images after each image registration to refine the estimated camera poses and triangulated point positions.
- A more efficient BA parameterization for dense photo collections using redundant view mining.
- **Scene Graph Augmentation:** COLMAP uses a multi-model geometric verification strategy to augment the scene graph with the appropriate geometric relation.

The incremental SfM pipeline used in COLMAP is show in the Figure 2.1

2.3 Experiments using COLMAP

This section shows the results of using the improved version of the conventional SfM method, i.e. COLMAP followed by multi-view stereo for dense depth map estimation. The original data is in the form of videos captured by an endoscope using phantom organs. Using the MATLAB camera calibration toolbox, the parameters of the endoscope camera were determined. The video was then split into frames. Since the video involves both rotation and translation motion of the endoscope, successive frames can be viewed as images of a scene with overlap, captured from different camera poses.

The radial distortion is corrected using the estimated camera parameters. This is followed by the COLMAP pipeline, which includes feature extraction, feature matching and multi-view stereo for dense point cloud estimation. This pipeline is shown in Figure 2.2.

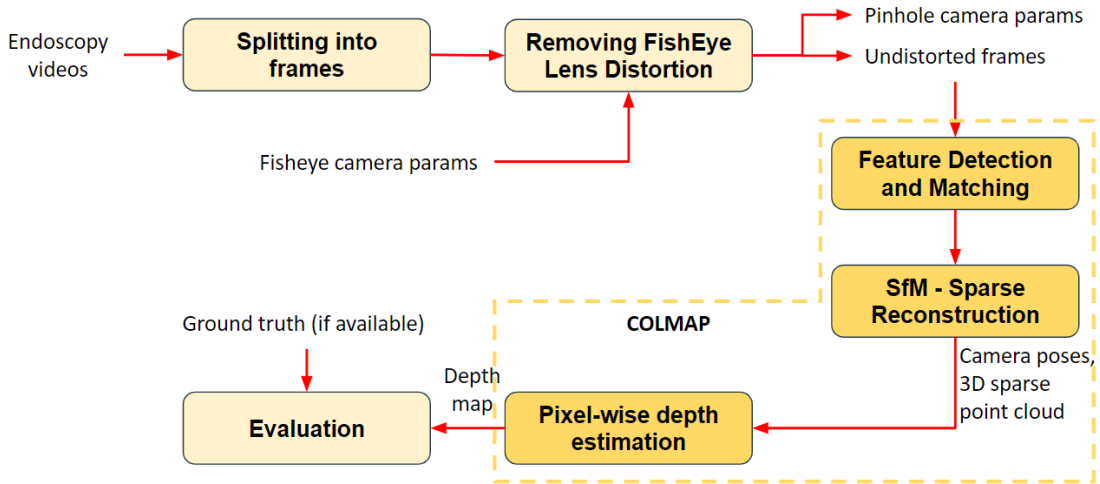


Figure 2.2: Depth Estimation Pipeline

On using the SIFT [9] algorithm for feature extraction, followed by the pipeline shown above, the results shown in Figure 2.3 were obtained. The first image shows the input colour image, and the second and third images are the predicted depth maps using geometric and photometric priors, respectively. The depth maps show multiple artefacts, and they are very noisy. This happens because endoscopic images have very few features in them. If the number of features detected by SIFT is very low, then the point cloud generated by the SfM pipeline is very sparse. This results in an inferior quality depth map. Since the conventional feature extraction method did not perform

well, a learning-based feature extraction method SuperPoint [5] and a learning-based feature matching method [6] were used. The following sub-sections elaborate on these two methods.

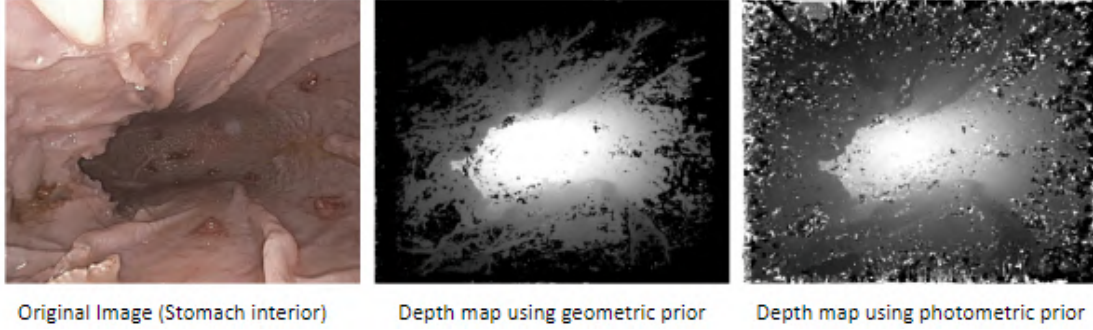


Figure 2.3: Depth map reconstructed by COLMAP

2.4 SuperPoint: Self-Supervised Interest Point Detection and Description

SuperPoint [5] is a self-supervised deep learning method for feature extraction. It is a fully convolutional deep neural network trained to detect interest points and compute their accompanying descriptors. The method comprises of three main steps that are shown in Figure 2.4 First, in the Interest point pre-training step, a labelled synthetic dataset is used to train a base feature detector. This synthetic dataset has geometric shapes with unambiguous corner points, which are used to train the base detector. In the second step, interest point self-labelling, a homographic adaptation pipeline is used to warp each image, and the trained base detector is used to detect the features in each of these warped images. The warping needed to transform the original image into any of the warped images is known. The interest points detected in this step become the pseudo-ground truth interest points. In the third step, joint training, the SuperPoint network is trained using the pseudo-ground truth interest points and descriptors.

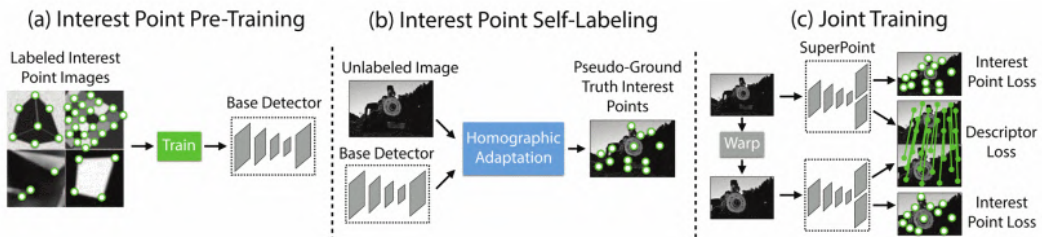


Figure 2.4: SuperPoint self-supervised training method

In the final joint training step, we take a pair of images where the relative warping is known and use the SuperPoint network to detect the interest point locations and descriptors. In this step, we get two outputs for each image. The interest point locations constitute one output, and the interest point loss can be calculated using the pseudo ground truth from the previous step. Using the descriptors, the interest points in the image pair can be matched. Since the warping is known, the ground truth matches are also known, and the feature descriptor loss can also be calculated. The sum of the interest point loss and descriptor loss can be minimised to train the SuperPoint network.

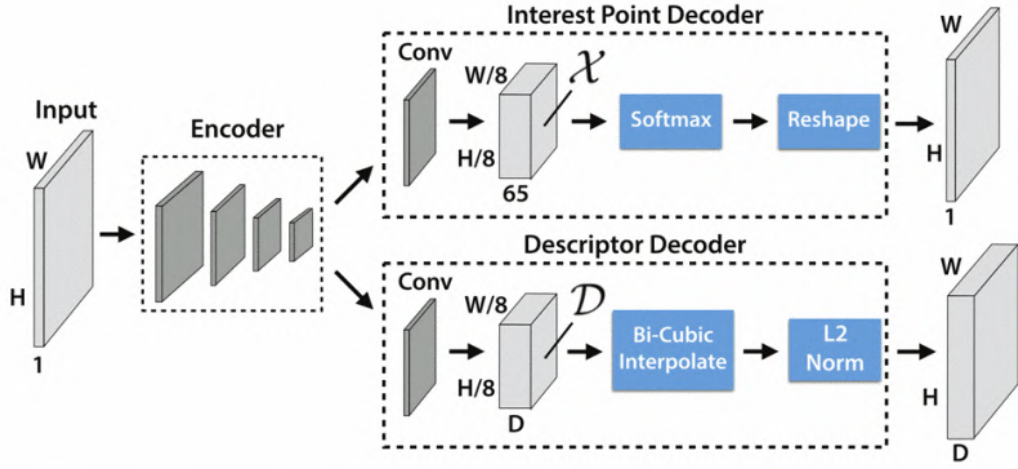


Figure 2.5: SuperPoint network architecture

2.5 SuperGlue: Learning Feature Matching With Graph Neural Networks

SuperGlue[6] is a neural network that matches two sets of local features by finding feature correspondences and rejecting non-matchable points. The SuperGlue network is made up of two main components: an Attentional Graph Neural Network and an Optimal Matching Layer.

The Attentional Graph Neural Network increases the distinctiveness of keypoints by integrating contextual cues such as spatial and visual relationship with other co-visible keypoints. A keypoint encoder maps keypoint positions and their visual descriptors into a single vector. The attention mechanism is used to relate the keypoints with their

surrounding contextual cues. Attentional aggregation builds a dynamic graph between key-points. Self-attention can attend anywhere in the same image, and is thus not restricted to nearby locations. Cross-attention attends to locations in the other image, such as potential matches that have a similar appearance.

The optimal matching layer creates an M by N score matrix and then finds the optimal partial assignment using the Sinkhorn algorithm. As in the standard graph matching formulation, the partial assignment can be obtained by computing a score matrix for all possible matches and maximizing the total score. This is equivalent to solving a linear assignment problem. The SuperGlue network architecture is shown in Figure 2.6

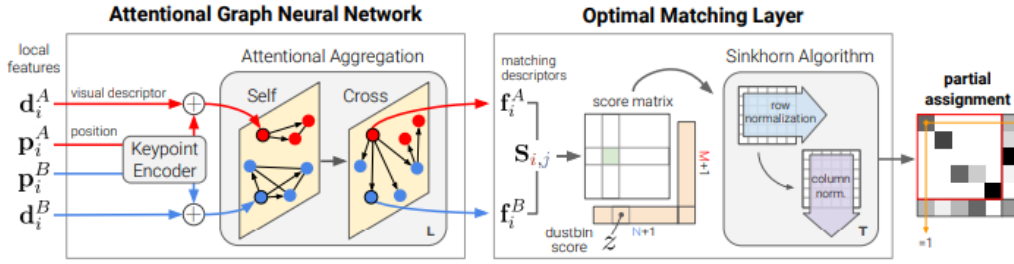


Figure 2.6: SuperGlue network architecture

2.6 Experiments using SuperPoint features and Super-Glue matching algorithm

This section shows the results obtained on using the SuperPoint network for feature extraction. For feature matching, the nearest neighbour matching and the SuperGlue methods have been contrasted. The rest of the pipeline following the feature matching step is the same as shown in Figure 2.2.

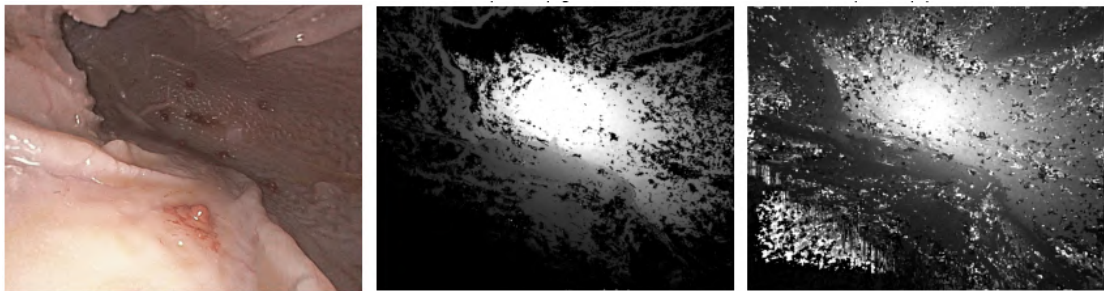
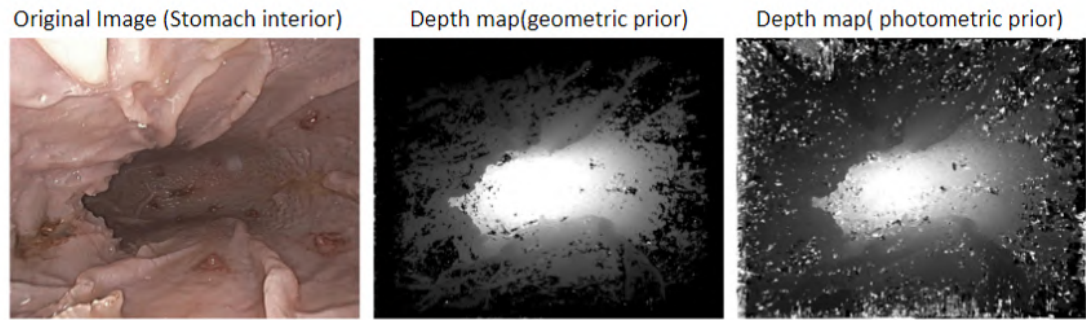


Figure 2.7: Using SuperPoint features and nearest-neighbour matching (endoscopic images)

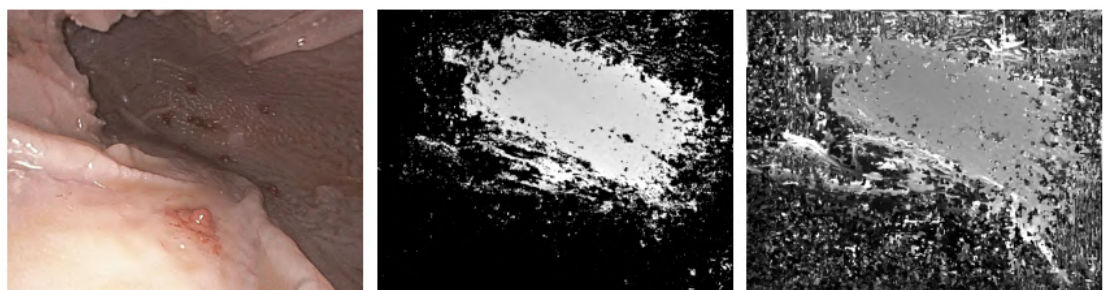
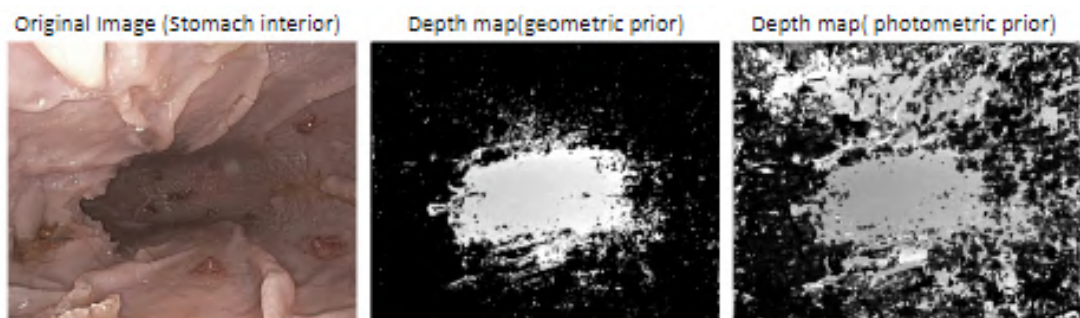


Figure 2.8: Using SuperPoint features and SuperGlue for matching (endoscopic images)

Figure 2.7 shows the result obtained on using SuperPoint for feature extraction and the nearest neighbour matching algorithm. Figure 2.8 shows the result obtained on

using SuperPoint for feature extraction and the SuperGlue network for matching. It is observed that the depth maps are still very noisy. Using SuperGlue for matching results in even poorer quality depth maps as compared to the nearest neighbour approach.

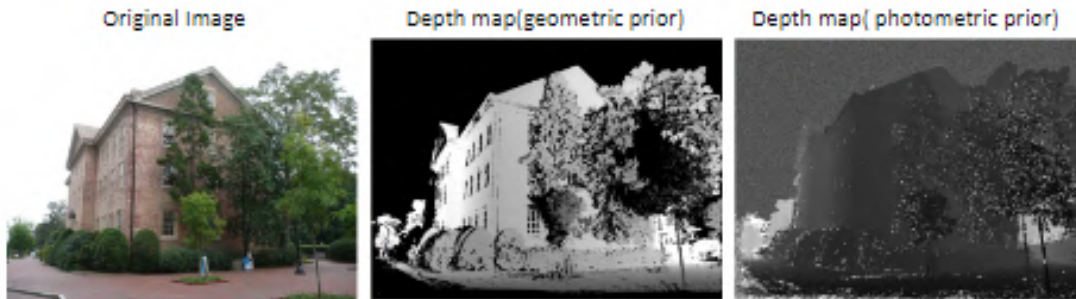


Figure 2.9: Using SuperPoint features and nearest-neighbour matching

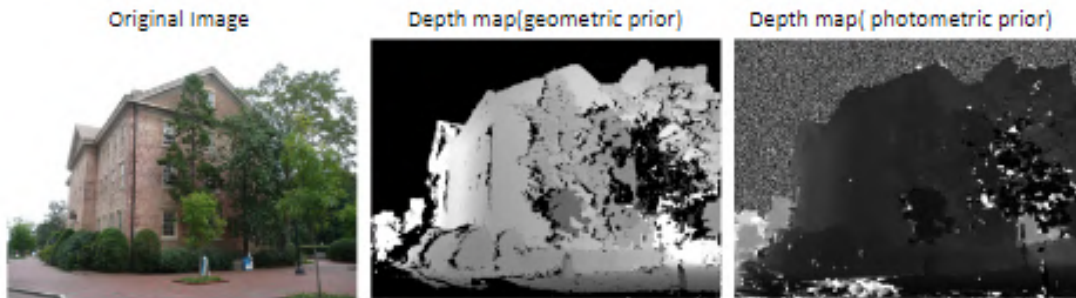


Figure 2.10: Using SuperPoint features and SuperGlue for matching

The above methods were tested on some natural images, and the results are shown in Figure 2.9 and 2.10. Since these methods are working well for natural images, it indicates that the poor performance can be attributed to the fact that the SuperPoint and SuperGlue neural networks were not trained on endoscopic images.

The above experiments on endoscopic images were repeated multiple times, and it was observed that the number of SuperPoint features detected and the number of matches found by the SuperGlue network are identical across runs, but the results (depth maps) vary a lot. This variation is because the number of features detected is very low. The number of high confidence matches detected by SuperGlue is also quite low. These issues arise because the models have not been trained on endoscopic images. The smooth and texture-less nature of the endoscopic images also causes feature extraction methods to fail. In order to overcome the dependence on the feature detection and matching steps, given that the texture is very less, the next approach is to use deep learning methods for Structure from Motion.

CHAPTER 3

Unsupervised Learning Approach

3.1 Introduction

In the previous chapter, I presented the results using a conventional approach to Structure-from-Motion. Deep learning-based methods, namely SuperPoint and SuperGlue, were used for feature extraction and matching, but there was little improvement. This is because these methods still depend on the detection and matching of features. However, an endoscopic image is largely texture-less and does not have very distinct features. In order to overcome the dependence on the feature detection and matching steps, given that the texture is very less, the next approach is to use deep learning methods for Structure from Motion.

It is very difficult to get real-world endoscopic images due to the various constraints involved in recording and using medical data. Moreover, obtaining the depth information for each pixel is a huge challenge. In the absence of ground truth depth, it is not possible to train a supervised learning algorithm to predict depth. In order to overcome this problem of paucity of data, we can use a simulated dataset in environments like Unity or Blender. In such simulation software, it is possible to control the camera parameters, the motion of the camera inside the organs, capture scene information like absolute depth for every pixel etc. In this chapter, we explore the EndoSLAM dataset [1], which contains real-world and simulated endoscopic images. For the simulated images, we also have ground-truth depth information.

In the absence of ground-truth depth information for real-world endoscopic images, unsupervised learning methods can be used to train a neural network to predict scene depth. This chapter also presents EndoSfM-Learner[1], an unsupervised monocular depth and pose estimation method which combines the idea of residual networks with a spatial attention model which directs the network to focus on distinctive and highly-textured regions of endoscopic images.

3.2 Self-supervised Monocular Depth Estimation

Per-pixel ground-truth depth is very difficult to obtain at a large scale to train a neural network [10]. Self-supervised learning is a promising approach to training depth estimation neural networks in order to overcome this limitation. In conventional methods like SfM or multi-view stereo, a minimum of two views are needed for triangulation. In monocular depth estimation, the aim is to be able to obtain a dense depth image from a single colour image. The results obtained on using the Monodepth2 [10] network on endoscopic images are shown in Figure 3.1. The model is trained on the EndoSLAM dataset. Therefore the results for endoscopic images are poor. However, we can see that the regions with higher depth appear darker and the regions closer to the camera have a lighter colour.

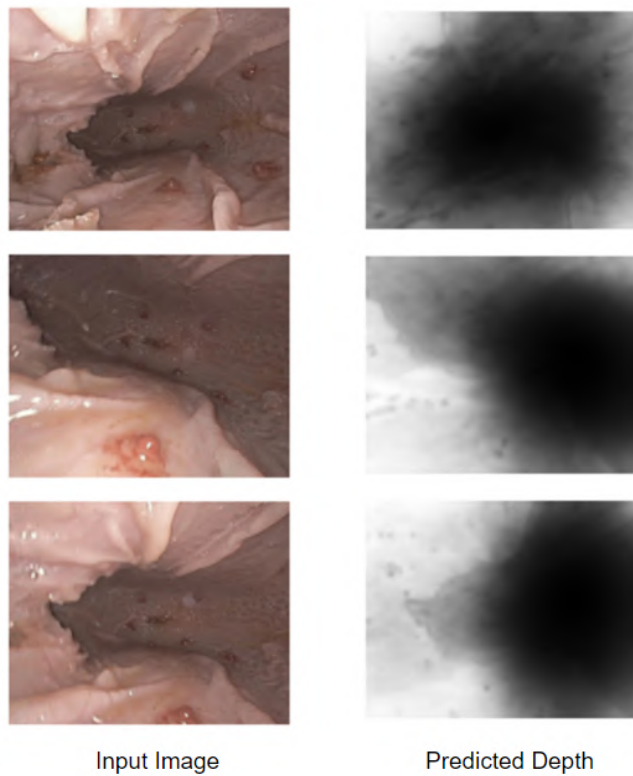


Figure 3.1: Pixel-wise depth predicted by Monodepth2 network

3.3 Endo-SfMLearner

Endo-SfMLearner [1] is a spatial attention-based monocular depth and pose estimation neural network. This network applies spatial attention modules to residual networks,

which helps the network focus on distinguishable regions with a lot of texture. Endoscopy images have some specific problems like frequently changing lighting conditions (which stems from the shallow depth of field), scale inconsistency between consecutive frames, organ tissues exhibiting non-Lambertian reflectance properties which reflect light non-diffusely etc. The proposed approach makes use of a brightness-aware photometric loss which improves the robustness to rapid illumination changes from one frame to the next that are commonly seen in endoscopic videos.

3.3.1 Working of the network:

EndoSfM Learner jointly trains a depth estimation network and a pose estimation network from an unlabelled endoscopic dataset. The depth estimation network is called DispNet and the pose estimation network is called Attention PoseNet. The depth estimation network predicts depth from a single image, and the Attention PoseNet predicts the relative pose between two frames.

On considering two images I_i and I_{i+1} , the DispNet predicts the depth for the two images, D_i and D_{i+1} respectively. I_i and I_{i+1} are concatenated and given as input to the Attention PoseNet, which predicts the relative pose. Using this relative pose, the depth map D_i is warped to the frame of image I_{i+1} to get the warped depth D'_{i+1} . The geometric consistency loss is computed between D'_{i+1} and D_{i+1} . The inconsistency between the projected depth map D'_{i+1} and the estimated depth map D_{i+1} is minimized to jointly train DispNet and Attention PoseNet. This is illustrated in the Figure 3.2.

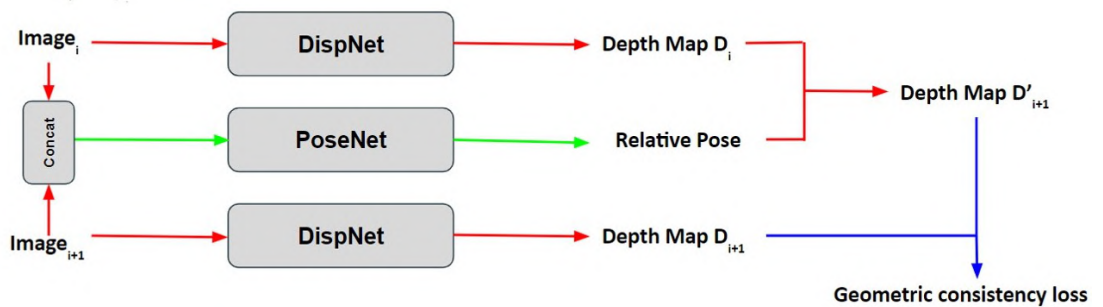


Figure 3.2: Working of the Endo-SfMLearner network

3.3.2 Training objective and Loss function

There are three loss functions used to train the network. They are brightness-aware photometric loss, smoothness loss, and geometry consistency loss. The method uses an affine brightness transformation between successive frames to overcome problems that arise due to the brightness constancy assumption. First of all, the new reference image, \hat{I}_i , is synthesised via interpolating I_{i+1} . The difference stemming from the illumination changes between consecutive frames might mislead the network. We propose to equate the brightness conditions between these two images as a robust way of supervising the training phase.

Changing the distance between the organ tissue and the camera leads to scale inconsistency between consecutive frames. This issue is solved by including geometric consistency loss. It confirms that D_i provides the same scene under the transformation of D_i by predicted relative poses $P_{i,i+1}$.

The overall objective of the method is to minimize the weighted sum of the brightness-aware photometric loss L_{bp}^M , smoothness loss L_s and geometry consistency loss L_{GC} as follows:

$$L = \alpha L_{bp}^M + \beta L_s + \gamma L_{GC}$$

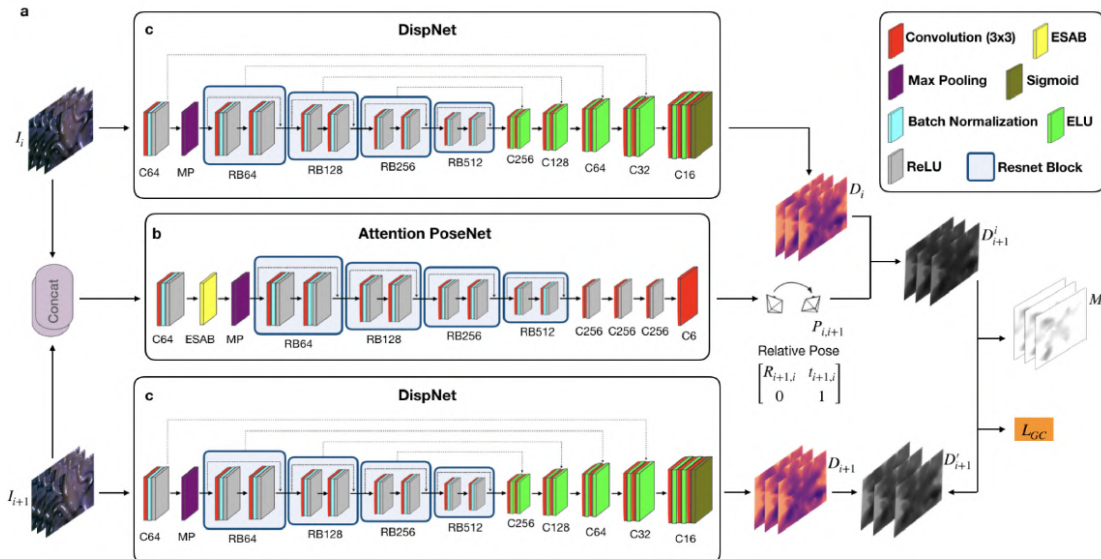


Figure 3.3: Endo-SfMLearner network architecture

3.3.3 Endo-SfMLearner spatial attention block (ESAB)

The intuition behind the ESAB module in encoder layers is to guide the pose estimation network (Attention PoseNet) by emphasising texture details and depth differences of pixels. Spatial attention selects a specific region of the input image, and features in those regions are processed by the attention block.

3.4 EndoSLAM dataset

The EndoSLAM [1] dataset is a dedicated endoscopic dataset designed for the development of pose estimation and dense 3D map reconstruction methods. This dataset is recorded using multiple endoscope cameras and ex-vivo porcine gastrointestinal organs belonging to different animals. The real-world images, however, do not have pixel-wise ground-truth depth information.

In addition to the experimentally collected real-world data, the EndoSLAM dataset includes synthetically generated data using a 3D simulation environment Unity. This synthetic dataset has ground-truth information for pixel-wise depth and also for the camera poses. This dataset can be used for domain adaptation and transfer learning. Since deep neural networks need a large amount of data for training, a large amount of synthetically generated endoscopic images can be used to train the models and then these models can be used to perform the depth map estimation task on real-world endoscopic images. Research in recent years has shown that large amounts of synthetic data can improve the performance of learning-based vision algorithms and can ameliorate the difficulty and expense of obtaining real data in a variety of contexts.

3.5 Experiments

3.5.1 Depth Estimation:

The Endo-SfMLearner was trained on a dataset that consists of the OlympusCam images (from the EndoSLAM dataset) and the images collected using our endoscope and phantom models at the HTIC lab. The original images are radially distorted and they

need to be rectified using the radial distortion parameters. The successive frames have very less variation. In order to increase the amount of variation between the successive frames, the frames are sampled and every fifth frame is selected. The images are then arranged into sequences of length 3. There are a total of 890 sequences, of which the training and validation split is done in the ratio: 90% training data and 10% validation data. The shape of the images is (1080 x 1350 x 3). The training is done on a GPU. The network is trained for 40 epochs with randomly shuffled batches and each sequence contains 3 consecutive images, optimize by ADAM with an initial learning rate 0.0001 and validate after each epoch. While testing the model, a single image is sufficient to predict the depth map. The results are shown in Figure 3.4.

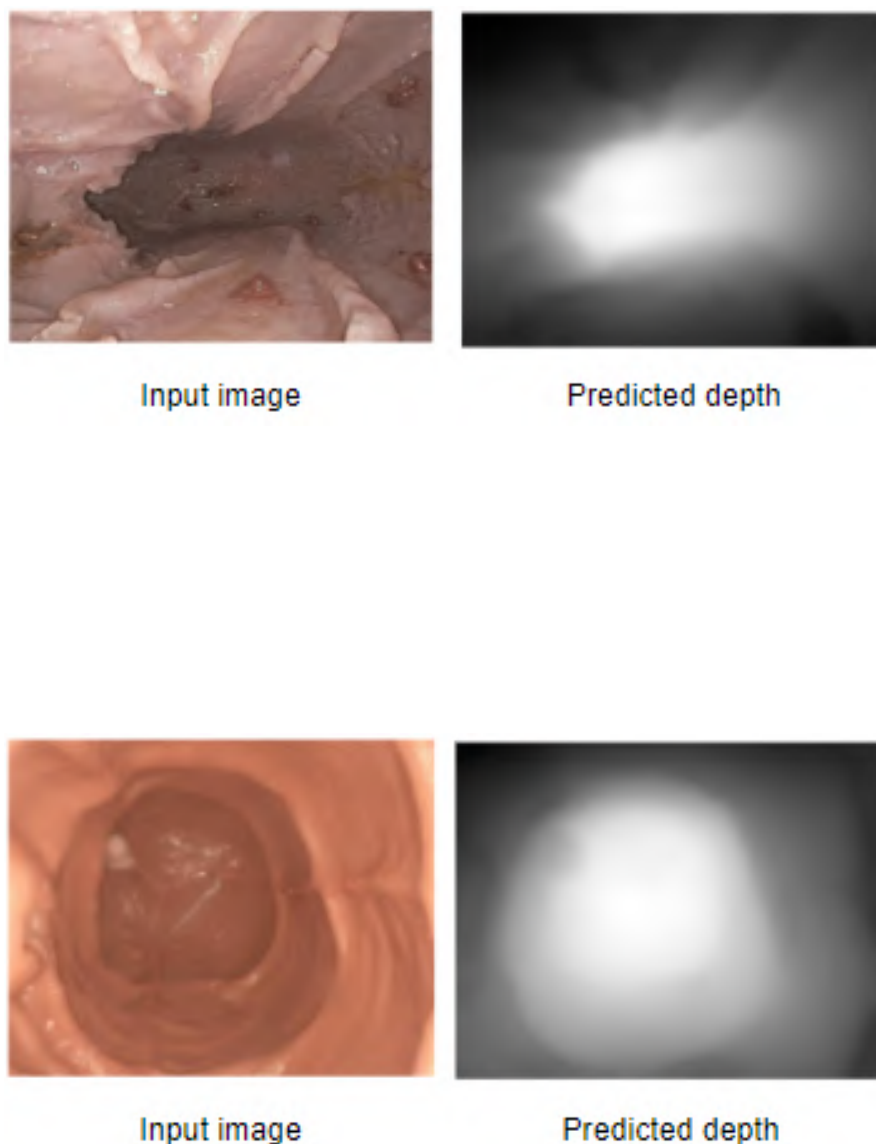
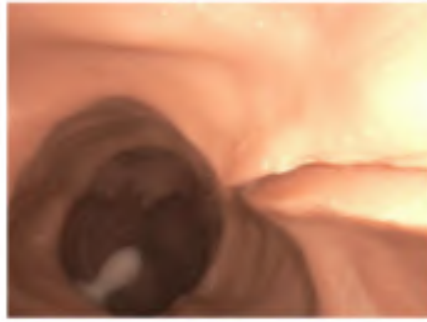
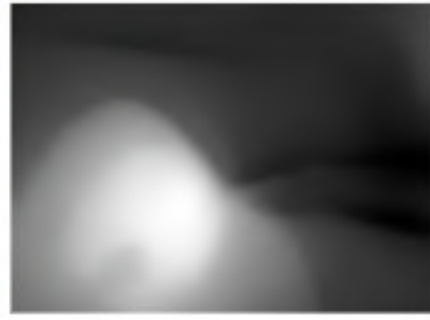


Figure 3.4: EndoSLAM Depth Predictions for model trained on real data



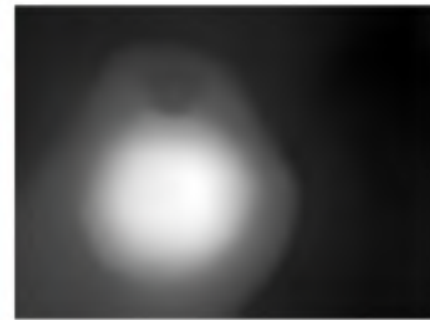
Input image



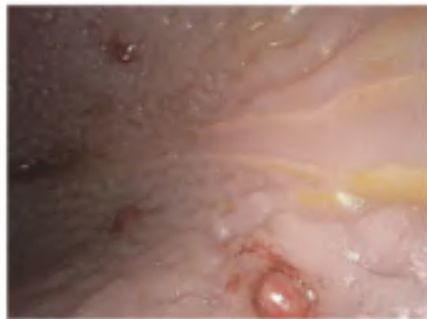
Predicted depth



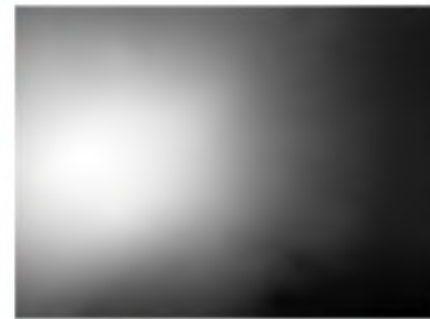
Input image



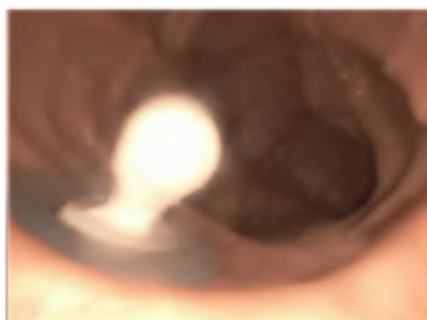
Predicted depth



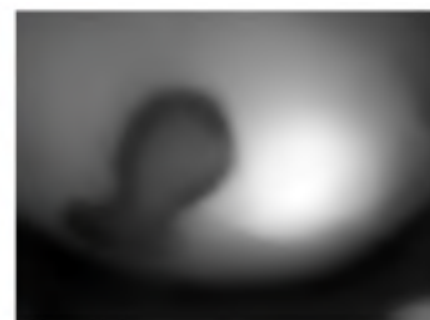
Input image



Predicted depth



Input image



Predicted depth

Figure 3.5: EndoSLAM Depth Predictions for model trained on real data

Then, the Endo-SfMLearner was trained on the synthetic EndoSLAM dataset. The dataset contains endoscopic images of a colon, stomach and small intestine. There are 1257 small intestine images, 900 colon images and 300 stomach images. The original images are radially distorted and they need to be rectified using the radial distortion parameters. The images are then arranged into sequences of length 3.

There are a total of 1440 sequences, of which the training and validation split is done in the ratio: 90% training data and 10% validation data. The shape of the images is (320 x 320 x 3). The training is done on a GPU. The network is trained for 100 epochs with randomly shuffled batches and each sequence contains 3 consecutive images, optimize by ADAM with an initial learning rate 0.0001 and validate after each epoch. It took 15 hours 43 minutes to train the model. While testing the model, a single image is sufficient to predict the depth map. The results are shown in Figure 3.6 and Figure 3.7.

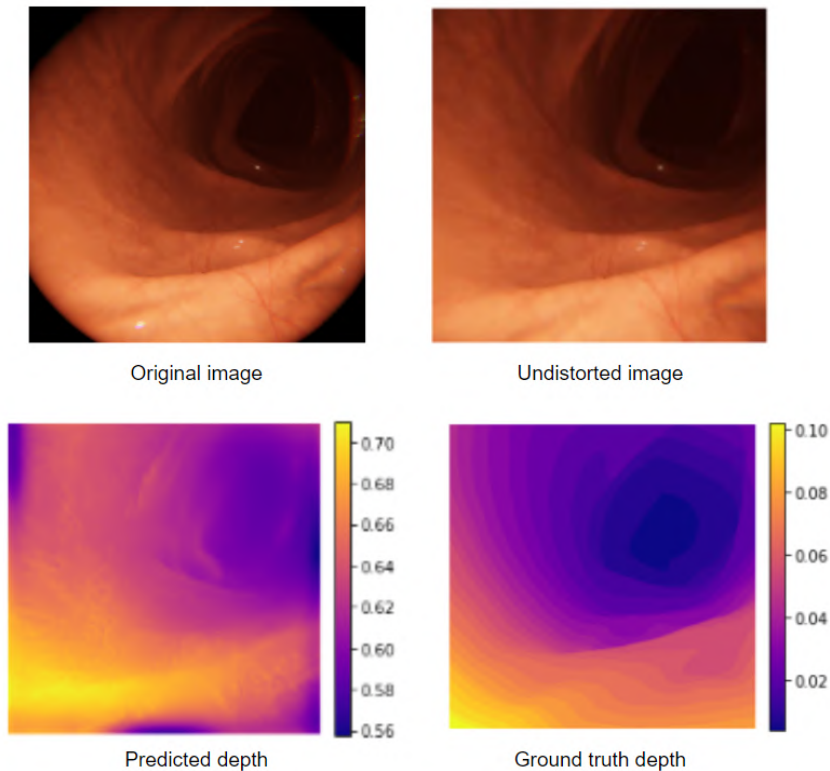


Figure 3.6: EndoSLAM Depth Predictions for model trained on simulated data

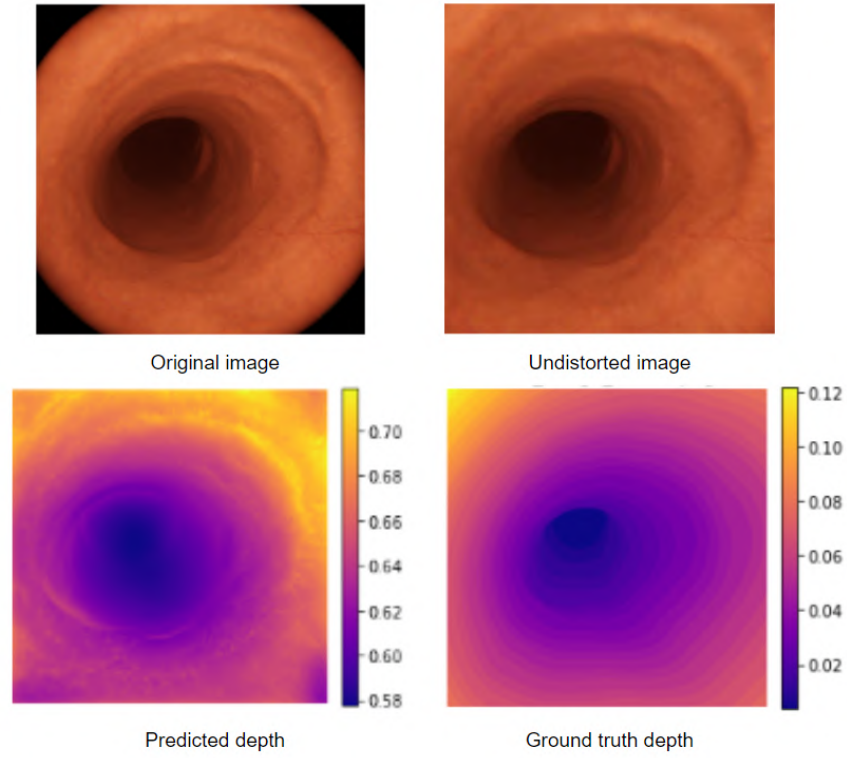


Figure 3.7: EndoSLAM Depth Predictions for model trained on simulated data

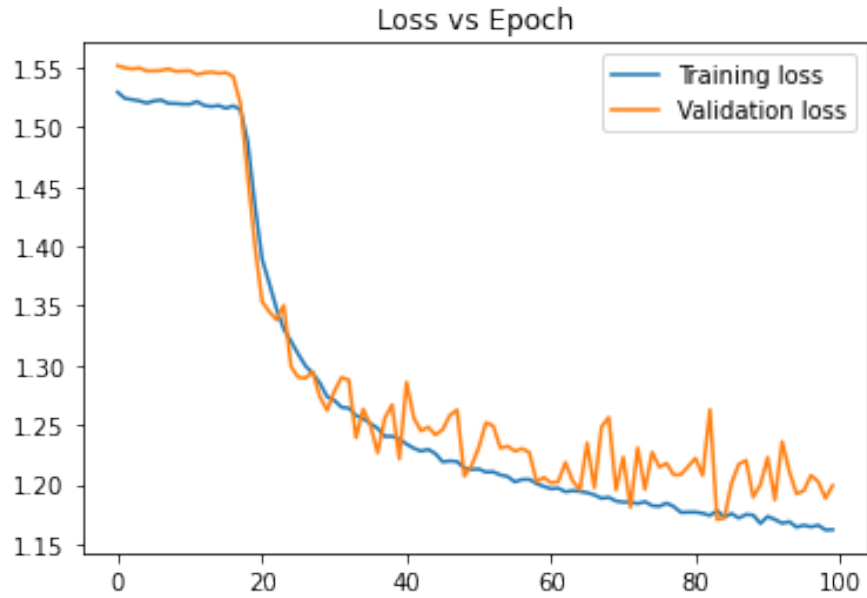


Figure 3.8: Training and Validation Loss vs Epoch for Endo-SfMLearner

3.5.2 Pose Estimation:

The Attention PoseNet estimated relative pose of the second frame with respect to the first frame. In this section, the pose estimation results are shown. Since the Endo-

SfMLearner training involves a joint training of both the depth and pose estimation network, no separate training is needed for pose estimation. We find the relative pose of each frame with respect to the previous frame. Let the Two camera positions at adjacent time instants $k - 1$ and k be related by the transformation $T_{k,k-1}$:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

where $R_{k,k-1}$ is the rotation matrix and $t_{k,k-1}$ is the translation vector. The current pose C_n can be computed from the pose of the previous frame C_{n-1} and the transformation as follows:

$$C_n = C_{n-1}T_{n,n-1}$$

Likewise, we can also write:

$$C_{n-1} = C_{n-2}T_{n-1,n-2}$$

On combining the above equations, we get:

$$C_n = C_{n-2}T_{n-1,n-2}T_{n,n-1}$$

On continuing this, till C_0 , pose of the n^{th} frame with respect to the first frame, i.e., the absolute pose can be calculated. This is a naive method to obtain the absolute pose estimate from the relative poses. We can improve the estimates using Pose-Graph optimization [11]. The camera poses can be represented as a pose graph, which is a graph where the camera poses form the nodes and the transformations (like $T_{k,k-1}$) between the camera poses form the edges between nodes. Any additional transformation i.e. relative pose estimate can be added as a constraint to the optimization by adding it as an edge between the corresponding node frames. Let e_{ij} be the edge constraints between nodes i and j . The cost function is:

$$\sum_{e_{ij}} \|C_i - T_{e_{ij}}C_j\|^2$$

Pose-Graph optimization aims to find the pose estimates that minimize this cost function.

In order to plot and compare the ground truth poses and the estimated poses it is important that both trajectories are transformed to the same scale. This is achieved using least squares estimation (Umeyama alignment) [12]. After this alignment the ground-truth trajectory and the estimated trajectory for the first 150 frames of the small intestine trajectory are plotted in Figure 3.9. The full trajectory for 1257 frames of the small intestine trajectory are plotted in Figure 3.10.

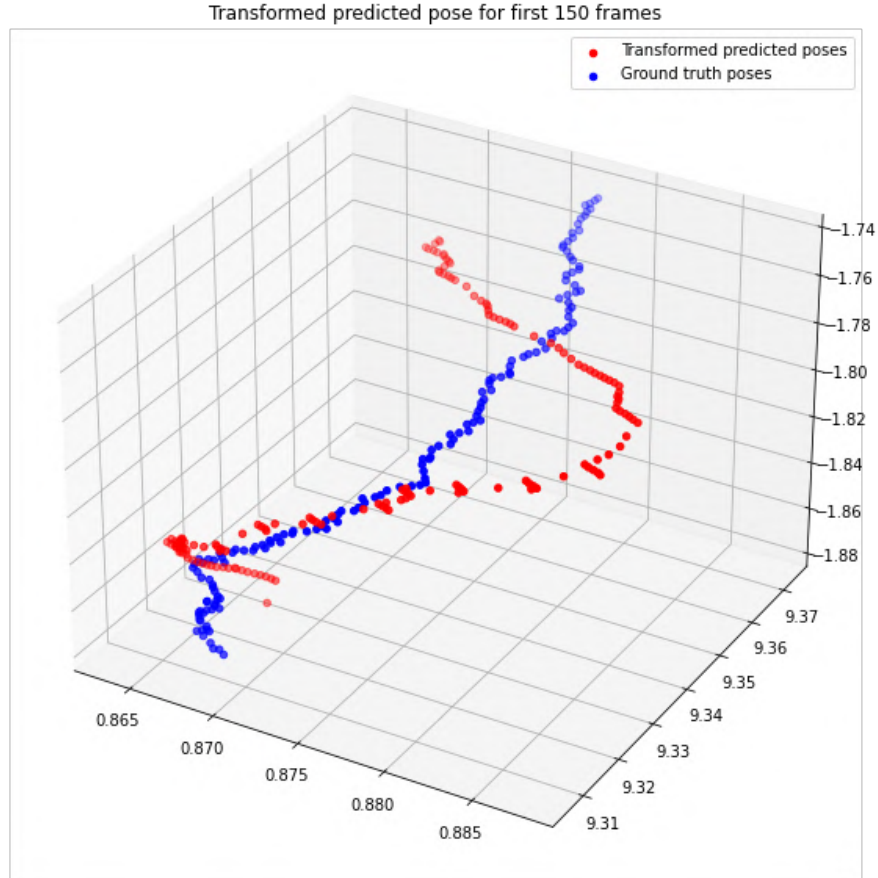


Figure 3.9: Estimated trajectory and ground-truth trajectory for first 150 frames of the Small Intestine dataset

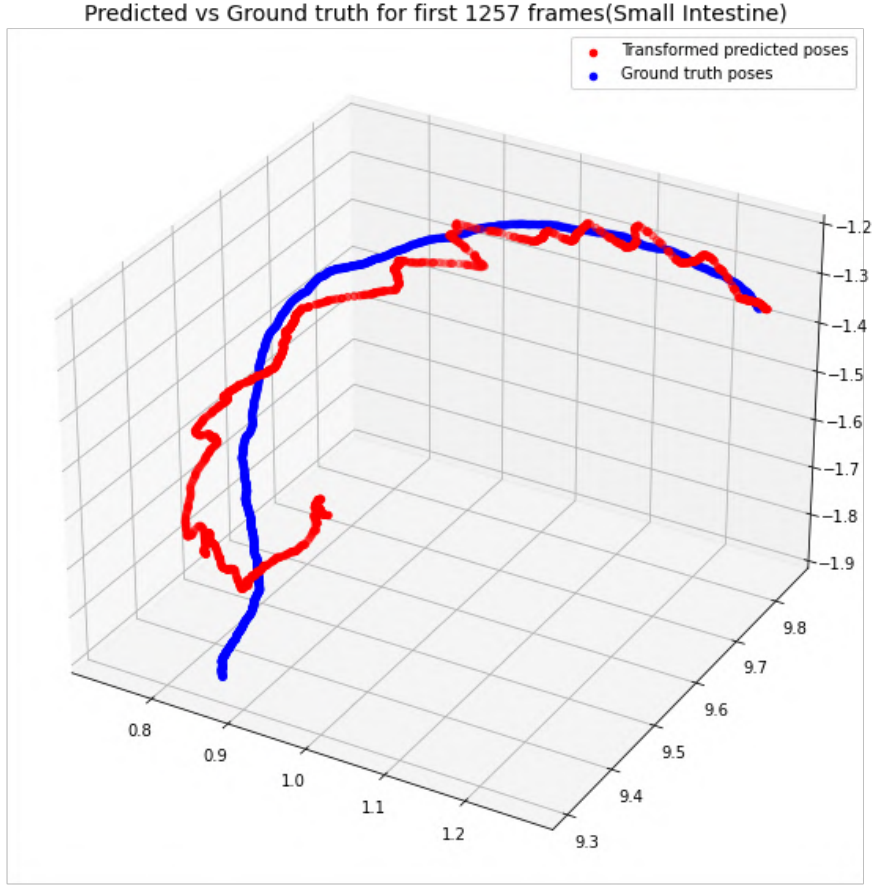


Figure 3.10: Estimated trajectory and ground-truth trajectory for the Small Intestine dataset

3.5.3 Error metrics

Table 3.1 shows the error metrics for the pose estimation. Absolute trajectory error (ATE) is the average deviation from the ground-truth trajectory per frame. It is calculated by taking the entire trajectory into consideration. It is the RMSE of the Euclidean distance between the predicted positions in the trajectory and the ground-truth.

RPE stands for relative pose error and it measure the error in the relative pose of two nearby frames. It compares the reconstructed relative transformation between nearby poses with the ground-truth transformation. It has two components, one is the error in the estimate of the relative rotation between the frames and the other is the error in the estimate of the relative translation. The RPE is computed for all pairs of frames in the trajectory and then the average value is considered.

	Small Intestine	Colon	Stomach
ATE (m):	0.01607	0.02153	0.0367
RPE (m):	0.00174	0.00219	0.00152
RPE (deg):	0.48175	0.34216	0.5992

Table 3.1: Error metrics for pose estimation using EndoSLAM dataset (For the first 150 frames of each organ’s trajectory)

	Small Intestine	Colon	Stomach
ATE (m):	0.09324	0.08246	0.05591
RPE (m):	0.00196	0.0034	0.00187
RPE (deg):	0.37441	0.37001	0.48471

Table 3.2: Error metrics for pose estimation using EndoSLAM dataset (Full trajectory)

3.5.4 Summary

In this section, the EndoSLAM dataset and the Endo-SfMLearner network for depth and pose estimation were introduced. From the results obtained for both the real-world dataset and the synthetic dataset, it is clear that the estimates are qualitatively better than the conventional method used in the previous chapter. Further improvement is possible with increase in size of the dataset by using more organ trajectories or through data augmentation. The unsupervised training approach helps us overcome the challenge of obtaining pixel-wise depth information for endoscopic images. However, these methods are still dependent on the features in the image and will work well only for highly-textured images. This is clear from the attention mechanism used for the pose estimation network which dictates the network to focus on highly distinguishable and textured regions. However, endoscopic images are by nature very smooth and devoid of texture and the number of distinctive features is too low for a correspondence-based method to work.

CHAPTER 4

Photometric Stereo Endoscopy

4.1 Introduction

Photometric stereo [13] is a computer vision technique to estimate the surface normals of a scene from multiple images of the same scene captured under different lighting conditions. The scene and the camera have to remain fixed while the images are captured. Photometric stereo has significant advantages over Structure-from-Motion:

- Photometric Stereo produces dense depth maps and surface normal maps as the values can be computed at every pixel, but in SfM only the feature points are triangulated.
- Photometric Stereo does not need feature detection and matching, so it can work well for endoscopic images which have limited texture.
- Structure-from-Motion also requires very strong assumptions on surface motion (e.g. rigid or periodic motion), and requires sufficient motion baseline.
- Photometric Stereo can compute 3D information from just a single colour image [4].

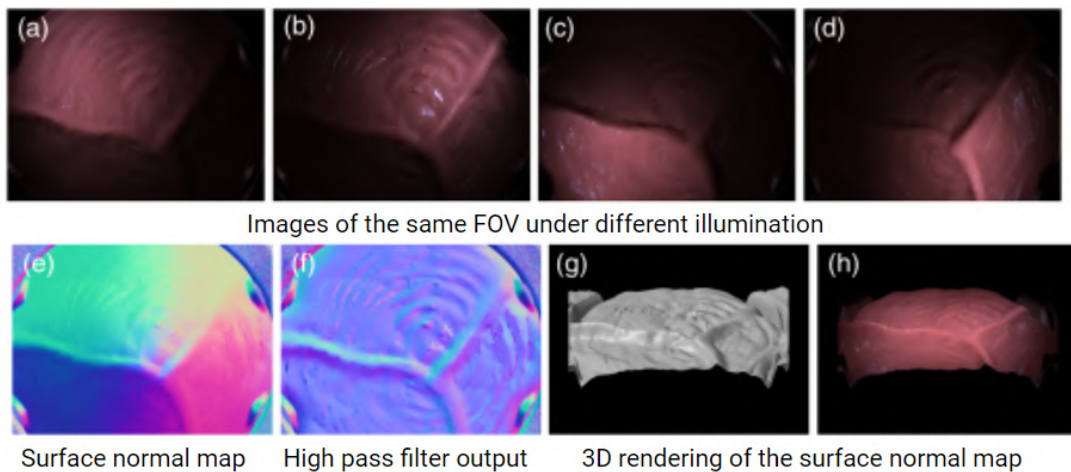


Figure 4.1: Conventional Photometric Stereo Example

4.2 Limitations of Conventional Photometric Stereo

In conventional photometric stereo, we assume a far-light scenario, i.e. the light source is at a large distance from the scene. Therefore, we can make the simplifying assumption that all the scene points receive light from the same direction and of the same intensity, as the depth variation within the scene is much smaller compared to the distance of the light from the scene. In the endoscopy setting, especially for in-vivo experimentation, the endoscopy probe (which has the light sources) is near to the scene being studied. Therefore, the far-light assumption does not hold. Another constraint in the endoscopy setting, due to size limitations, the photometric stereo setup cannot have large angular variations in the illumination directions because that results in a large spatial extent.

There is another constraint specific to endoscopic images. Since the diameter of the probe is limited by the diameter of the body's internal organs such as the small intestine, the separation between the light sources on the endoscope probe head is very low. The number of light sources is also limited due to less space available on the head of the endoscope probe. With fewer light sources, the quality of the reconstructions suffer. This effect can be observed on comparing Figure 4.6 and Figure 4.7. When we use just 3 light sources, the albedo is visibly dimmer and the surface normal estimates have a more greenish hue even on the front surface of the rabbit which should actually be more bluish.

4.3 Calibration Method

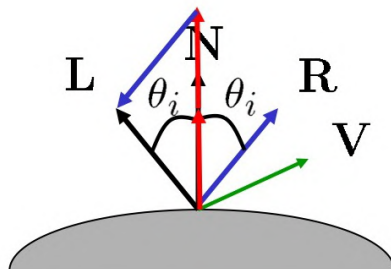


Figure 4.2: Light Direction Calibration

4.3.1 Theory

In this section, the method used for calibration of the light directions is explained. Figure 4.2 shows the surface of a sphere. L is the light source direction, N is the surface normal, R is the reflection vector. A highly specular object such as a chrome ball is used for calibration. First the centre and radius (r) of the chrome ball are determined using the MATLAB *imfindcircles* function. Let this be (x_c, y_c) . Next the specular region can be detected by selecting the pixels with intensity value greater than 0.9 times the maximum intensity value of the image. In photometric stereo, at a time, only one light source is switched on, therefore, there will be only specular highlight region on the sphere. The centre of this highlight can be calculated by taking a weighted average of the pixel coordinates in the highlight region, weighted by the pixel intensity values. Let the highlight centre be (x_h, y_h) . The reflection vector R for a specular highlight is the vector pointing perpendicularly out of the plane. Therefore, R is $(0, 0, -1)$. To calculate the surface normal vector $N(n_x, n_y, n_z)$:

$$n_x = \frac{x_h - x_c}{r}$$
$$n_y = \frac{y_h - y_c}{r}$$
$$n_z = \sqrt{1 - n_x^2 - n_y^2}$$

The light source direction for each image can be computed using the formula:

$$L = 2(N \cdot R)N - R$$

4.3.2 Lab Setup for Calibration and Data Collection

In order to test out the calibration method and to capture photometric stereo images of a few objects, the setup shown in Figure 4.3 was used. The setup uses 24 LEDs placed on a circular ring around the camera (Figure 4.4). Each LED is switched on, one at a time, and 24 images of the object are captured with the object and camera kept fixed.



Figure 4.3: Lab Setup for Calibration and Data Capture



Figure 4.4: Ring of 24 LEDs around the camera

4.4 Conventional Photometric Stereo

4.4.1 Theory

Let I be the image in the form of a matrix. Let L represent the light source direction for this image. Let n represent the surface normals and let k represent the albedo. Then we can write:

$$I = k(L.n)$$

We can set this up as an optimization problem as shown:

$$\min_{kn} \|I - k(L.n)\|^2$$

On solving we get:

$$kn = (L^T L)^{-1} L^T I$$

The magnitude of the LHS should give the albedo since the surface normals are unit vectors.

4.4.2 Experiments

Figure 4.5 shows 3 out of the 24 images collected using a specular ball with with lab setup shown in Figure 4.3. On using all these 24 images in the conventional photometric stereo method explained above, the surface normal and albedo are obtained as shown in Figure 4.6. When we use only 3 images, the results are shown in Figure 4.7. It is very clear that the performance of photometric stereo improves when a larger number of light sources are used. When we use just 3 light sources, the albedo is visibly dimmer and the surface normal estimates have a more greenish hue even on the front surface of the rabbit which should actually be more bluish.



Figure 4.5: 3 out of 24 calibration images of the specular ball

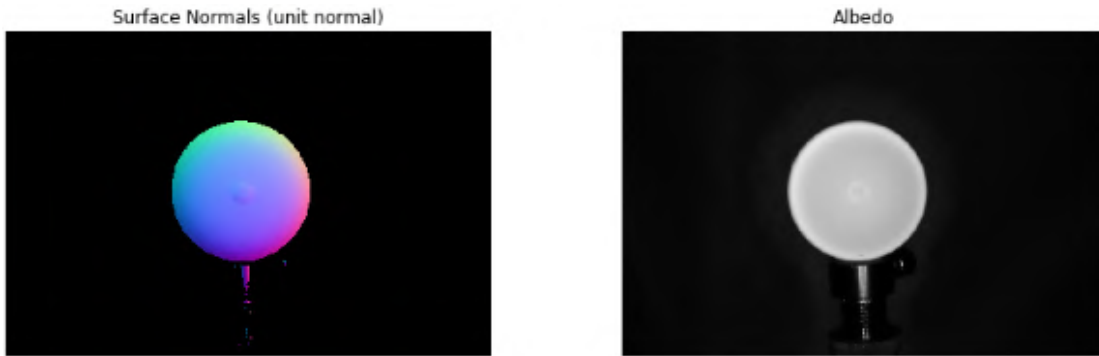


Figure 4.6: Photometric Stereo results on using all 24 images



Figure 4.7: Photometric Stereo results on using only 3 images

Figure 4.8 shows 3 out of the 24 images collected using a rabbit object having a Lambertian surface reflectance with with lab setup shown in Figure 4.3. On using all these 24 images in the conventional photometric stereo method explained above, the surface normal and albedo are obtained as shown in Figure 4.9. When we use only 3 images, the results are shown in Figure 4.10. It is very clear that the performance of photometric stereo improves when a larger number of light sources are used. When we use just 3 light sources, the albedo is visibly dimmer and the surface normal estimates

have a more greenish hue even on the front surface of the rabbit which should actually be more bluish.



Figure 4.8: 3 out of 24 images of the rabbit (Lambertian surface)



Figure 4.9: Photometric Stereo results on using all 24 images



Figure 4.10: Photometric Stereo results on using only 3 images

Using the Unity3D [14] software, some simulated photometric stereo images of a specular ball and the inside of a colon were obtained. In this case photometric stereo is done using only three images of each scene. This limitation arises from the need to

create a camera and light setup that can mimic an endoscope probe. A detailed explanation of the data collection method, the camera parameters, the light arrangements in the rendering environment etc. is given in Chapter 5.

Figure 4.11 shows the images of a specular ball rendered using Unity3D. Figure 4.13 shows the images of an endoscopic scene rendered using Unity3D. The surface normal images and the albedos are shown in Figure 4.12 and Figure 4.14 respectively.

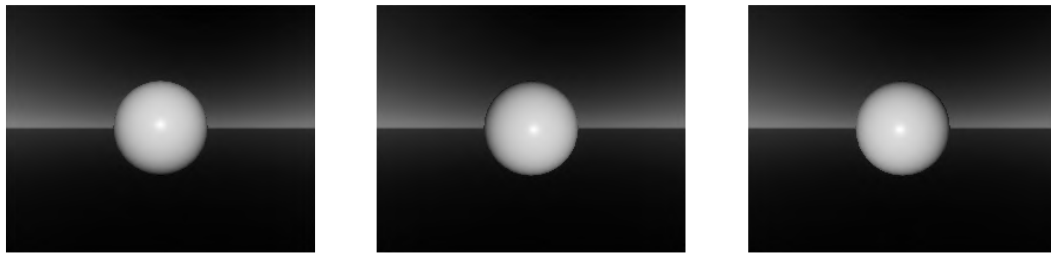


Figure 4.11: Photometric stereo images of a specular ball (simulated)

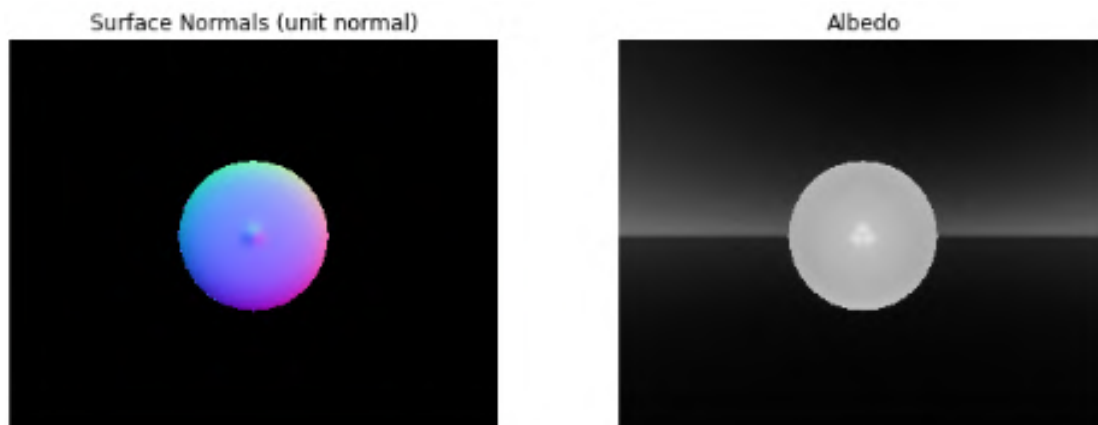


Figure 4.12: Photometric Stereo results for the rendered specular ball



Figure 4.13: Photometric stereo images of the colon (simulated)

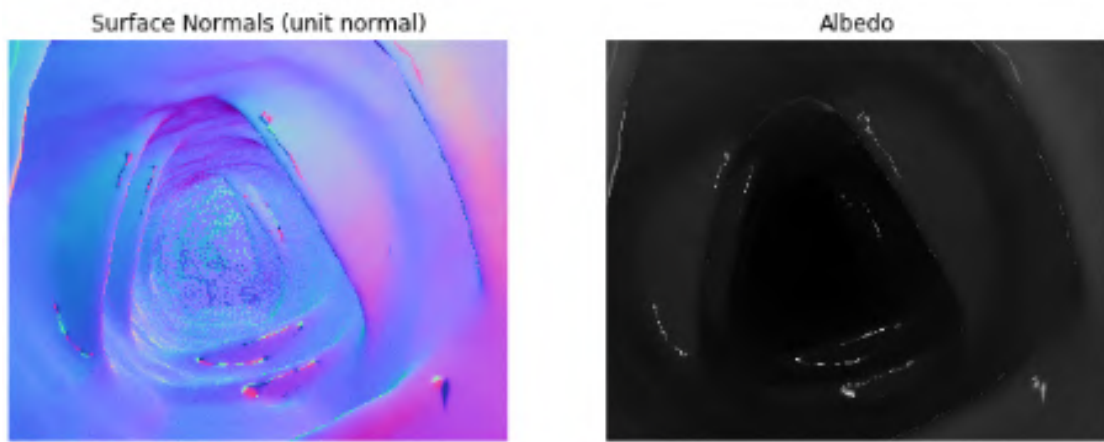


Figure 4.14: Photometric Stereo results for the rendered colon

There is no ground truth surface normal or depth information available for these images. Therefore, they have to be qualitatively evaluated. For the non-endoscopic scenes, such as the specular ball and the sphere, the predicted surface normals are very good and the albedo is also predicted correctly. However, for the endoscopic scene, we see a large number of artefacts in the surface normal image. Moreover, the portion of the scene facing the camera, must be bluish since the colour associated with the direction out of the plane is blue. However, the right side of the surface normal map has a reddish colour which is an incorrect prediction. The poor performance is because we are using very few (three) light sources. The other reason is that, for the non-endoscopic scenes, the object lies entirely within a small depth range so the light directions as seen by all the points of the scene are not very different, so even the conventional photometric stereo algorithm produces good results. However, in the case of endoscopy, the scene has a very large depth range and the camera is very close to the scene. So the light directions as seen by different pixels vary greatly. Therefore, the conventional photometric stereo

method which assumes that all the pixels in the image receive light from the same direction will not work well for endoscopic images.

CHAPTER 5

Rendering a Photometric Stereo dataset for Endoscopic Images

5.1 Introduction

The results in the previous chapter showed that while conventional photometric stereo worked well for images of a specular sphere, rabbit etc., it performed poorly on endoscopic data. I also tested a near-field photometric stereo approach for the endoscopic images, but the results were not good. Therefore, I decided to explore deep learning methods for photometric stereo. Deep learning methods require large amounts of data for training. However, no large dataset of photometric stereo images for endoscopic scenes exists. Capturing such data in the real-world requires major modifications to the existing endoscope, both in terms of the physical setup, such as the number of lights surrounding the camera and also the internal software that controls the working of these lights and image capture. As it is not possible to obtain real-world endoscopic images, I have created a simulated photometric stereo dataset using the Unity3D environment [14]. In this chapter, the data collection pipeline is explored in detail. The full procedure for the data collection and other details are provided in the appendix (Chapter 8). This simulated dataset can be used to train a near-field photometric stereo deep learning model and the model can then be used to predict the 3D information given real-world photometric stereo images.

5.2 Data Simulation

5.2.1 Rendered organs

In order to render endoscopic scenes, we first need organ models. The models of gastrointestinal organs, namely the stomach, small intestine and colon were obtained from

VR-Caps [15]. These models were imported into our Unity project and used as the scenes for capturing endoscopic images.

5.2.2 Camera and Lights Setup

Since there are no endoscopes currently that are designed to capture photometric stereo data, the real-world data for testing our deep learning model will be obtained by using a camera-lights setup designed in our lab. This camera will be used to capture images of ex-vivo tissues. The camera and lights setup is designed to mimic the head of the endoscope probe, with the camera in the centre, surrounded by lights. The camera used is a Raspberry Pi camera and we use LEDs for lights. The lights are controlled programmatically through a Raspberry Pi board. In the Unity environment, a camera-lights setup is created to mimic this lab setup. There is a camera in the centre (whose properties are set to match with the Raspberry Pi camera) surrounded by four LEDs. Each LED is switched on, one at a time and four images are captured for every scene. In the Unity3D environment, the movement of the camera inside the organ, the switching on and off of the LEDs, the image capture and storage are all handled through code written in the C-sharp programming language. The setup that will be used in the lab for collecting real-world images of ex-vivo tissue data is shown in Figure 5.1. The dimensions are shown in Figure 5.2.

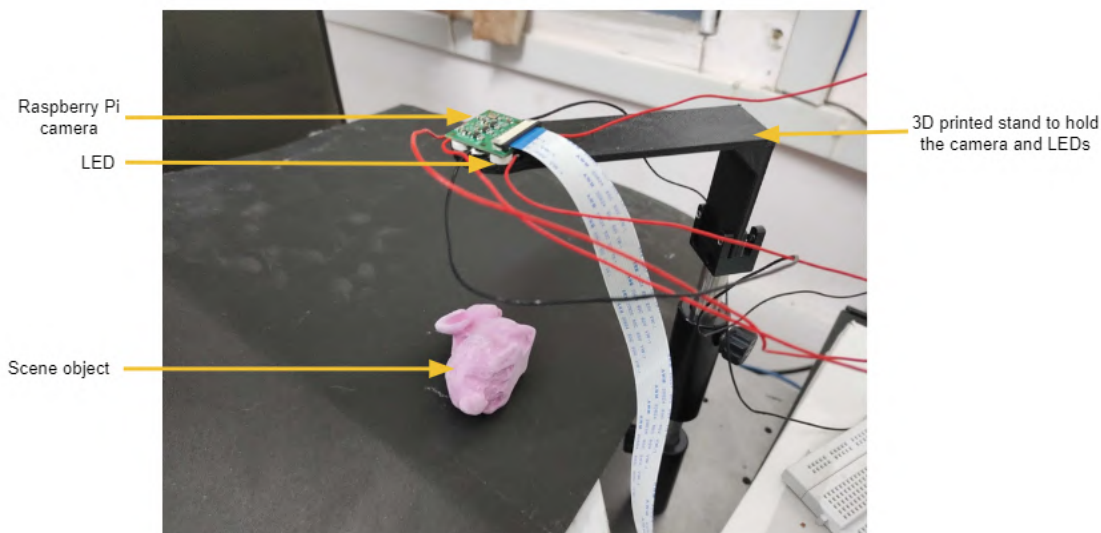


Figure 5.1: Lab setup using Raspberry Pi camera and LEDs

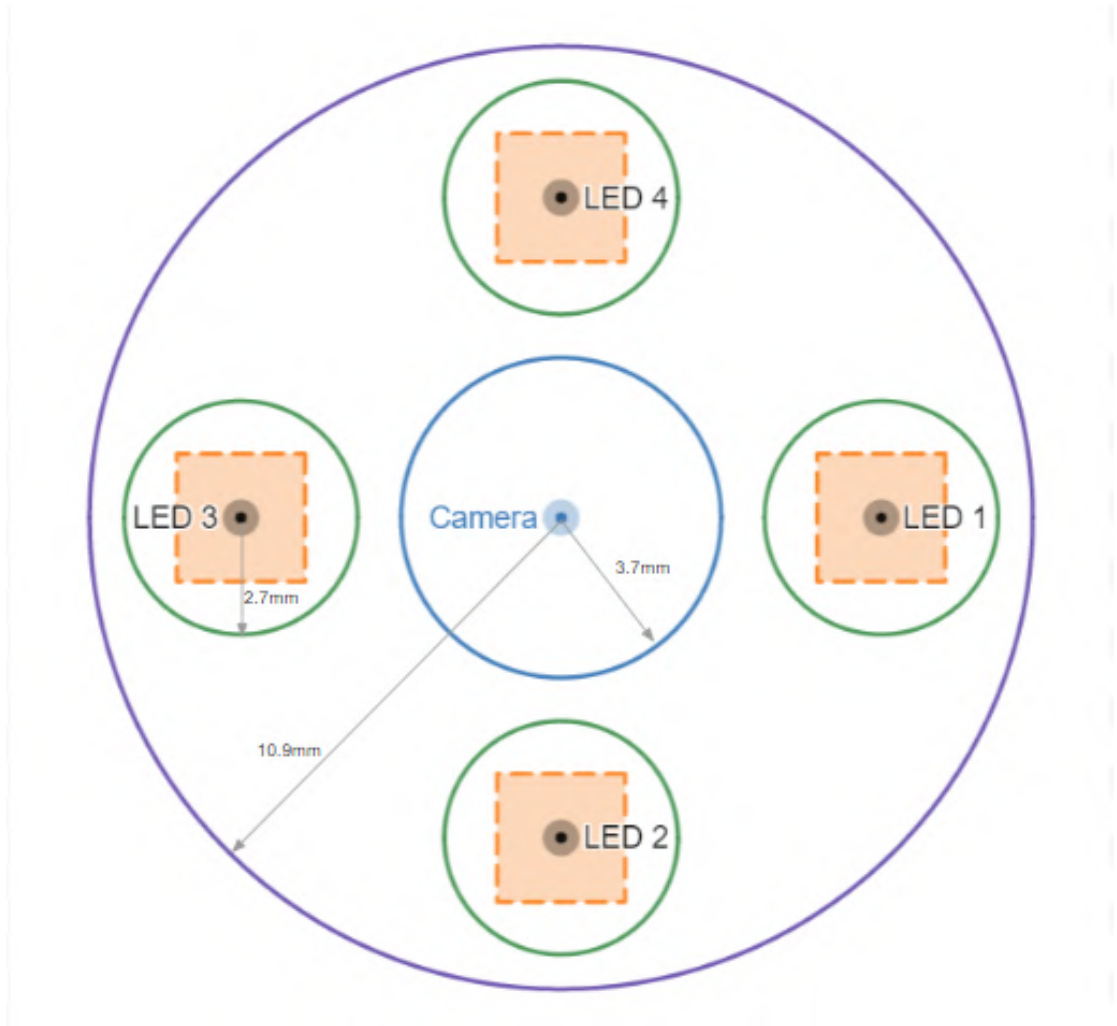


Figure 5.2: Diagrammatic representation of the Camera + LEDs model

In Figure 5.2, the big circle in the centre is for the camera. The other 4 smaller circles are for LEDs and the square in each circle is the region from which the LED positions are randomly sampled. The random sampling is done to make the neural network robust to errors in the light positions in the real 3D-printed model (i.e. the stand for holding the camera and LEDs).

5.3 Data Collection

The translation and rotation of the camera, switching on and off of the LEDs are controlled by the code. At each position of the camera, the orientation of the camera is changed 19 times. At each rotation, first we keep the LEDs at their fixed positions, then the positions are randomly sampled from within the orange square region shown in the

image. Since we have one fixed set of positions, and the sampling is done twice, at each rotation, we get 12 images of the scene. The Unity Image Sequence Recorder stores the image, and also the location and orientation of the camera, LEDs, both absolute and relative to the camera. The Unity AOV Recorder stores the depth and the surface normals. The dataset has 5244 scenes so far. Each scene comprises of 4 images captured with one LED switched on each time, pixel-wise depth map (both relative and absolute) and surface normal vectors.

5.3.1 Surface Normal Preprocessing

The Unity AOV Recorder stores the surface normals in the sRGB notation. To convert it into the RGB values that we normally use, we have to raise each pixel to the power 2.2.

$$x_p = r^{2.2}$$

$$y_p = g^{2.2}$$

$$z_p = b^{2.2}$$

r,g,b represent pixel intensity values in the original recorded surface normal image.

Now multiply each pixel value by 2 and subtract 1. It takes us from the 0-1 pixel intensities to the surface normals. Surface normal values range from -1 to 1.

$$X = (x_p * 2) - 1$$

$$Y = (y_p * 2) - 1$$

$$Z = (z_p * 2) - 1$$

This (X, Y, Z) is a surface normal vector with norm = 1. We can then do a normal mapping as follows:

X: -1 to +1 : Red: 0 to 255

Y: -1 to +1 : Green: 0 to 255

Z: 0 to -1 : Blue: 128 to 255

Unity stores the surface normals in world coordinates. Since we want the surface normals in the camera coordinate system, we have to multiply the surface normal vectors with the inverse of the 3D camera rotation matrix.

5.3.2 Getting Absolute Depth

Using Raycasting, a ray is sent from the camera centre, along the axis of the camera into the scene. This ray hits the scene at a point which forms the centre pixel of the final image. It is possible to record the point at which the ray strikes the scene and also the length of the ray from the origin at the camera centre to the point at which it hits the scene. This gives us the absolute depth of this point. The AOV Recorder stores the relative depth values. Since we know the absolute depth of one point, we can scale the other relative depths to get the absolute depths of the entire scene.

5.4 Examples

In Figure 5.3, the images of an endoscopic scene, captured with one light switched on at a time are shown. The simulation uses four light sources for every scene. Figure 5.4 shows the surface normal and depth maps for the same scene.

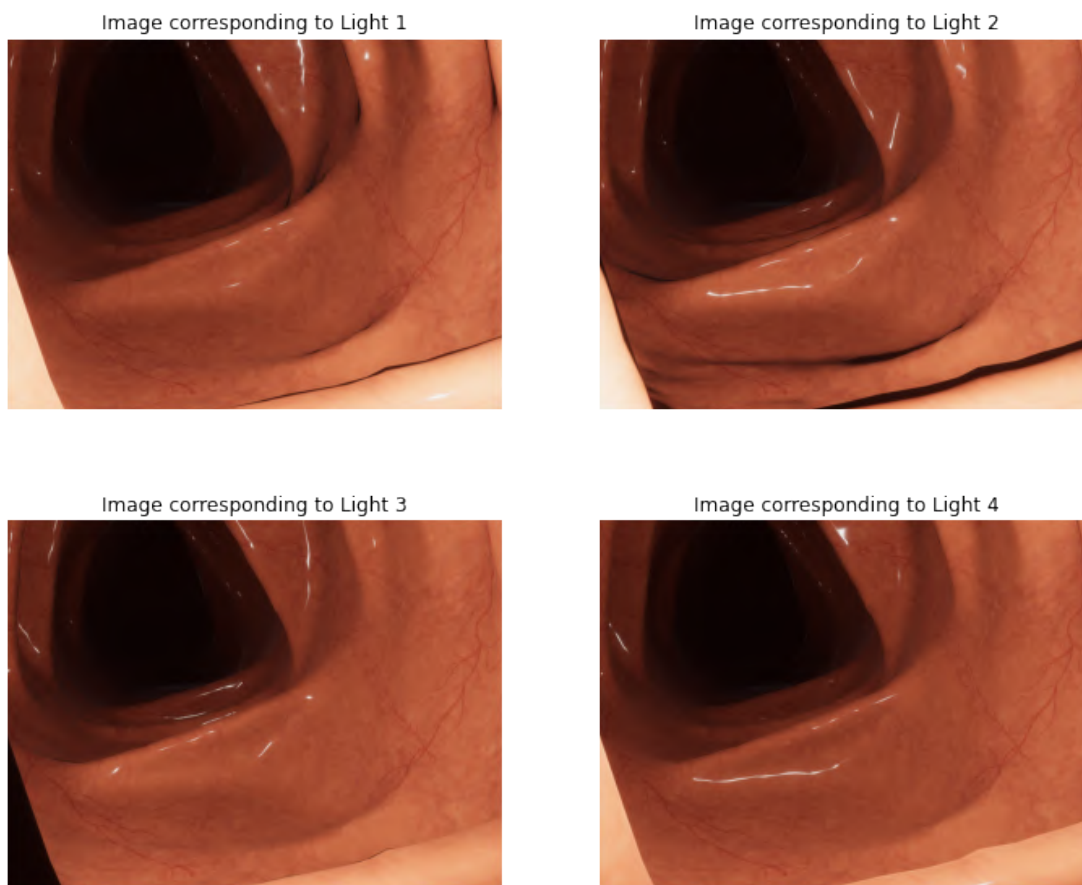


Figure 5.3: Images of an endoscopic scene with 4 light sources

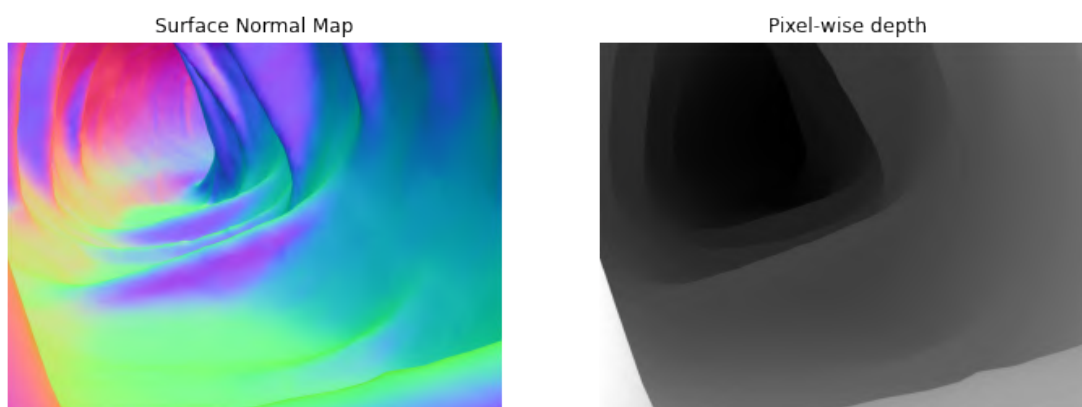


Figure 5.4: Surface Normal and Depth maps

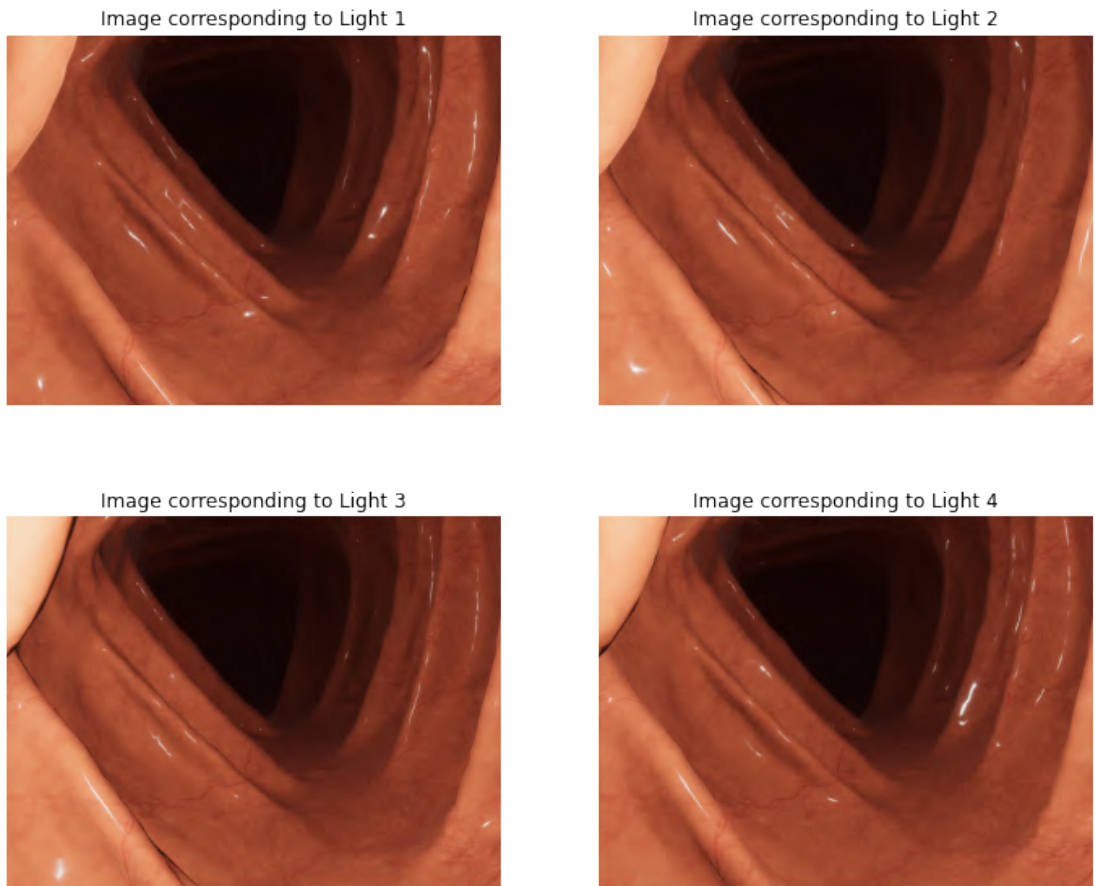


Figure 5.5: Images of an endoscopic scene with 4 light sources

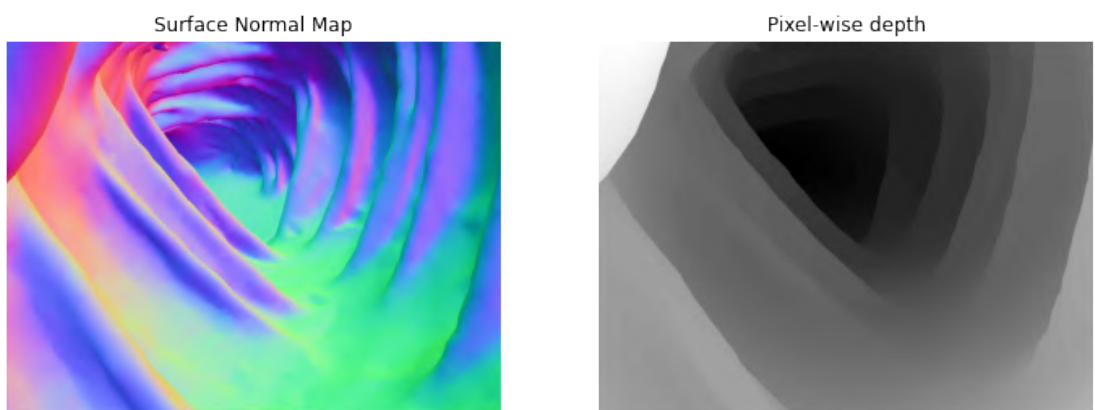


Figure 5.6: Surface Normal and Depth maps

Figure 5.7 and Figure 5.8 show the images, surface normals and depth maps for a scene where the organ has some polyps/lesions on it.

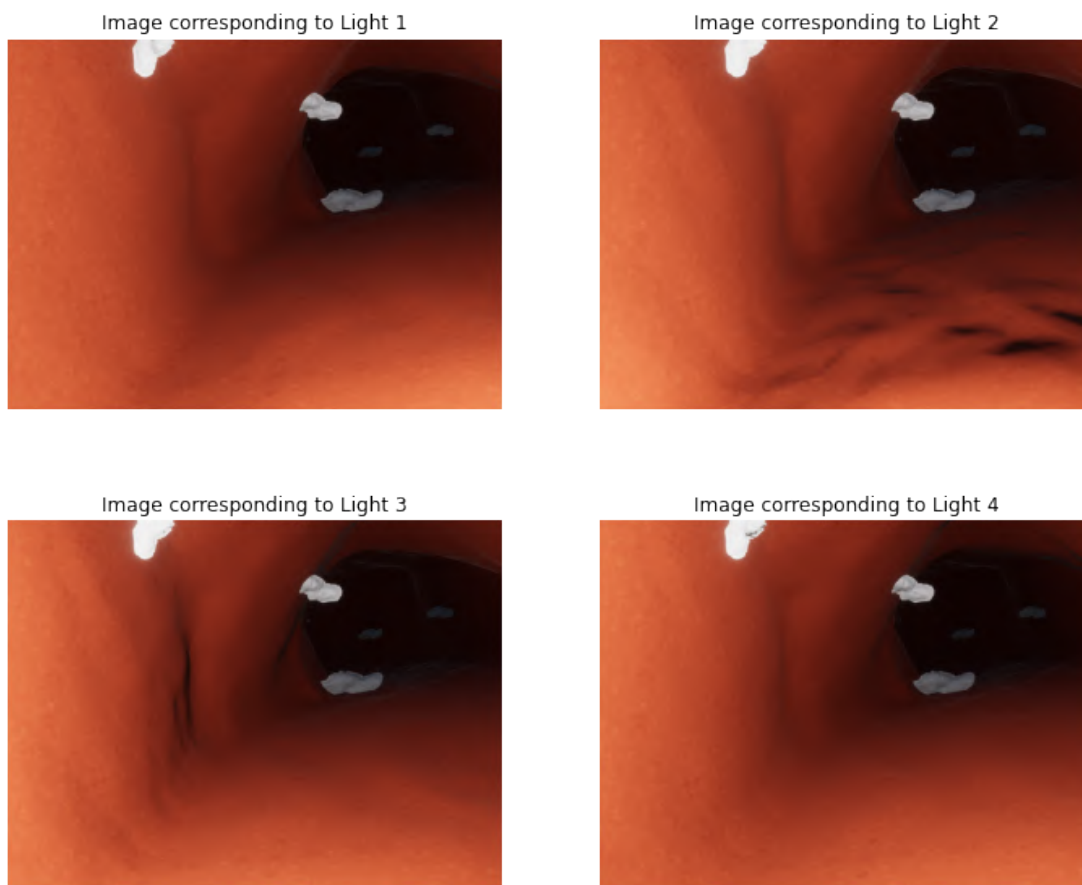


Figure 5.7: Images of an endoscopic scene with 4 light sources

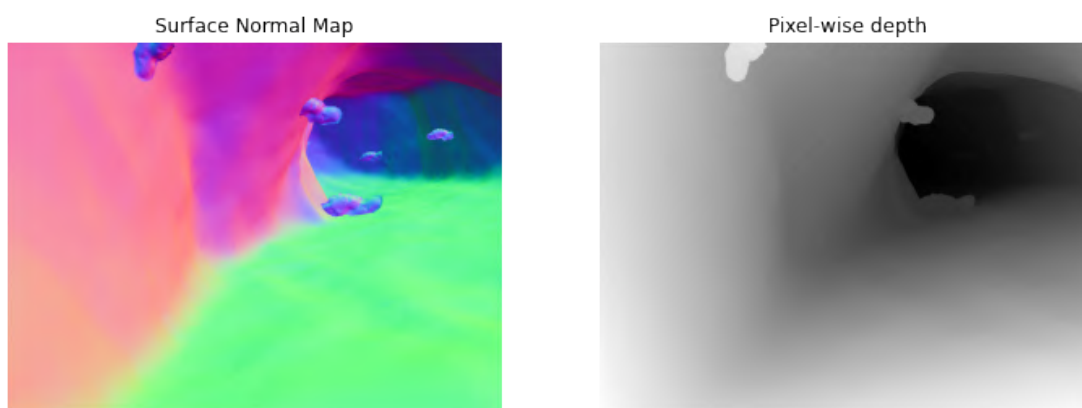


Figure 5.8: Surface Normal and Depth maps

CHAPTER 6

Deep Learning for Photometric Stereo

6.1 Introduction

In this chapter, a fast, light-weight, near-field, deep learning-based photometric stereo [16] approach is presented. Chapter 4 showed that both conventional and near-field photometric stereo methods did not work well for endoscopic scenes. Therefore, in this chapter, a learning-based solution will be presented. Due to the unavailability of a labelled photometric stereo dataset for endoscopic images that is large enough for training neural networks, the synthetic dataset explained in Chapter 5 is used for training the network.

6.2 Overview

The method involved training two recursive networks, one for predicted surface normals and the other is trained to predict pixel-wise depth maps.

The ‘per-pixel lighting’ (i.e., the relative lighting direction and attenuation factor for each pixel in the image) is estimated analytically by upsampling the predicted depth map from the previous step. Then, the surface normal for this scale is inferred given the input image, ‘per-pixel lighting,’ and estimated normal map from the previous scale. Finally, the depth map is predicted conditioned on the estimated normal map and the depth map from the previous scale. In each step of the recursion, the resolution is increased by a factor of 2 until it reaches the input image resolution. The number of steps in this recursion is decided by the resolution of the input image.

6.3 Experiments and Results

The pretrained model was tested with photometric stereo images captured in the lab and the results are shown in the subsequent images. Figure 6.1 shows the 3 out of the 24 input images of the rabbit object. Figure 6.2 shows the predicted surface normal map, depth map and the 3D structure as visualized in MeshLab [17].



Figure 6.1: 3 out of 24 images of the rabbit (Lambertian surface)

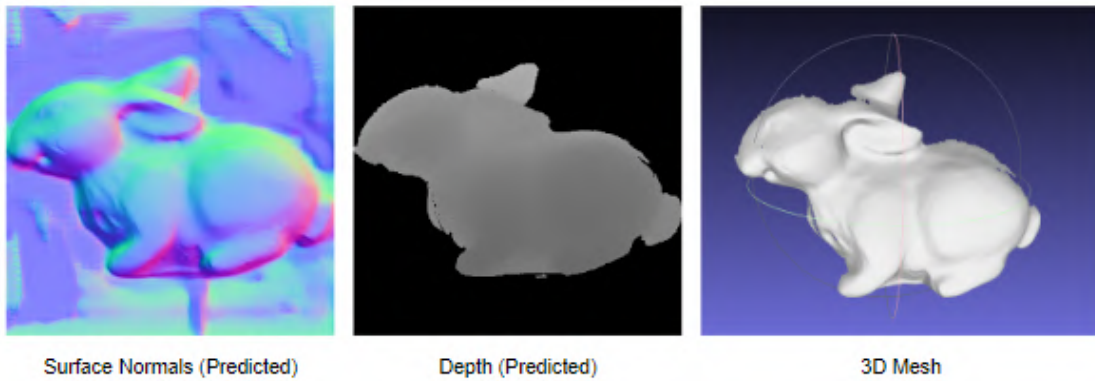


Figure 6.2: Predictions using the pretrained model (Fast Near-Field Photometric Stereo)

Figure 6.3 shows the 3 images of a sphere captured in Unity under different illumination conditions. Figure 6.4 shows the predicted surface normal map, depth map and the 3D structure as visualized in MeshLab [17].

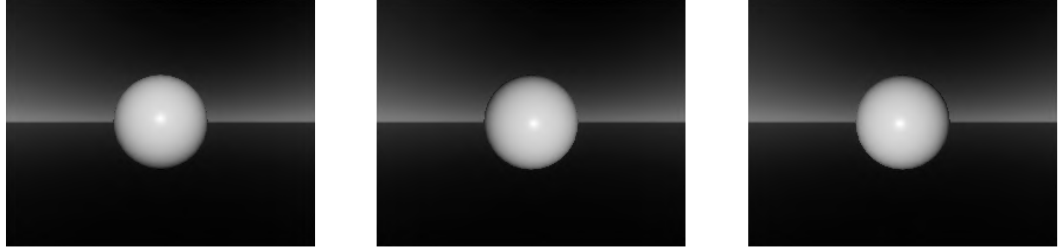


Figure 6.3: Images of the sphere (rendered in Unity) under different illumination conditions

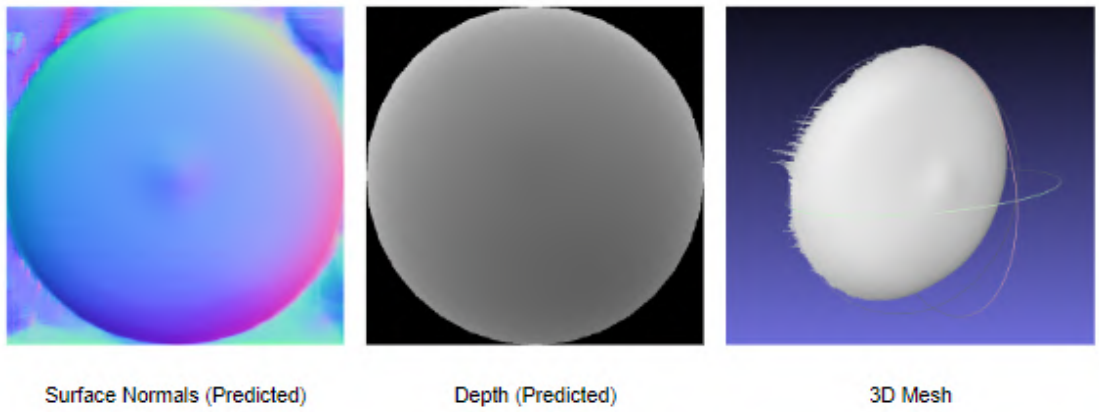


Figure 6.4: Predictions using the pretrained model (Fast Near-Field Photometric Stereo)

Figure 6.5 shows the 3 images of an endoscopic scene captured in Unity under different illumination conditions. Figure 6.6 shows the predicted surface normal map, depth map and the 3D structure as visualized in MeshLab [17].

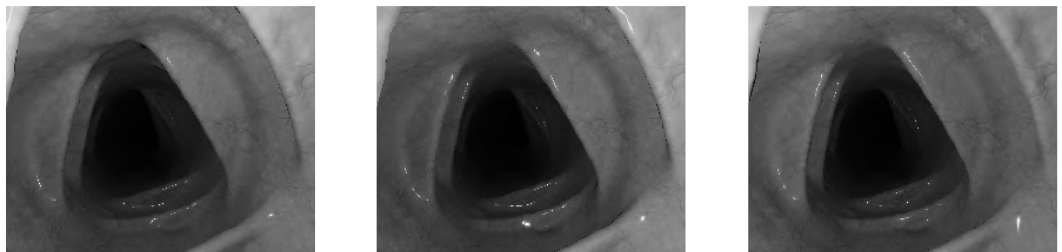


Figure 6.5: Images of the endoscopic scene (rendered in Unity) under different illumination conditions

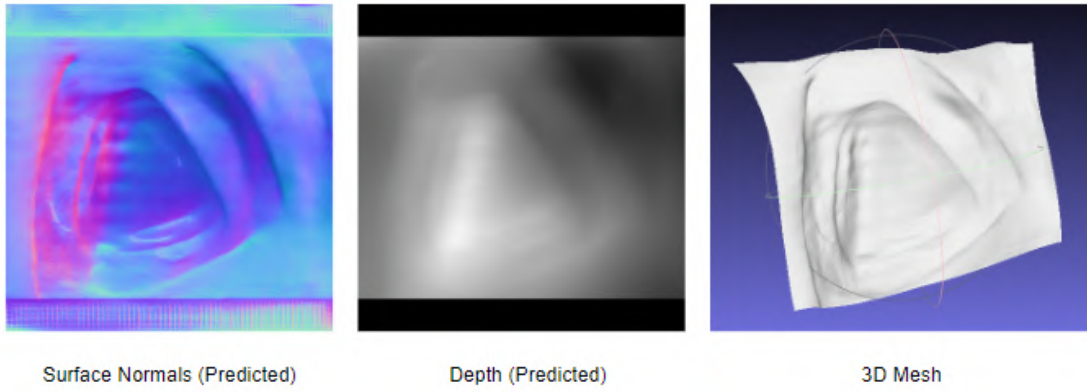


Figure 6.6: Predictions using the pretrained model (Fast Near-Field Photometric Stereo)

The fast, lightweight, near-field photometric stereo model was then trained on our synthetic which has 5244 endoscopic scenes. The training and test split is done in the ratio 90%:10%. The estimated surface normals and pixel-wise depth for some of the training images are shown in the subsequent figures.

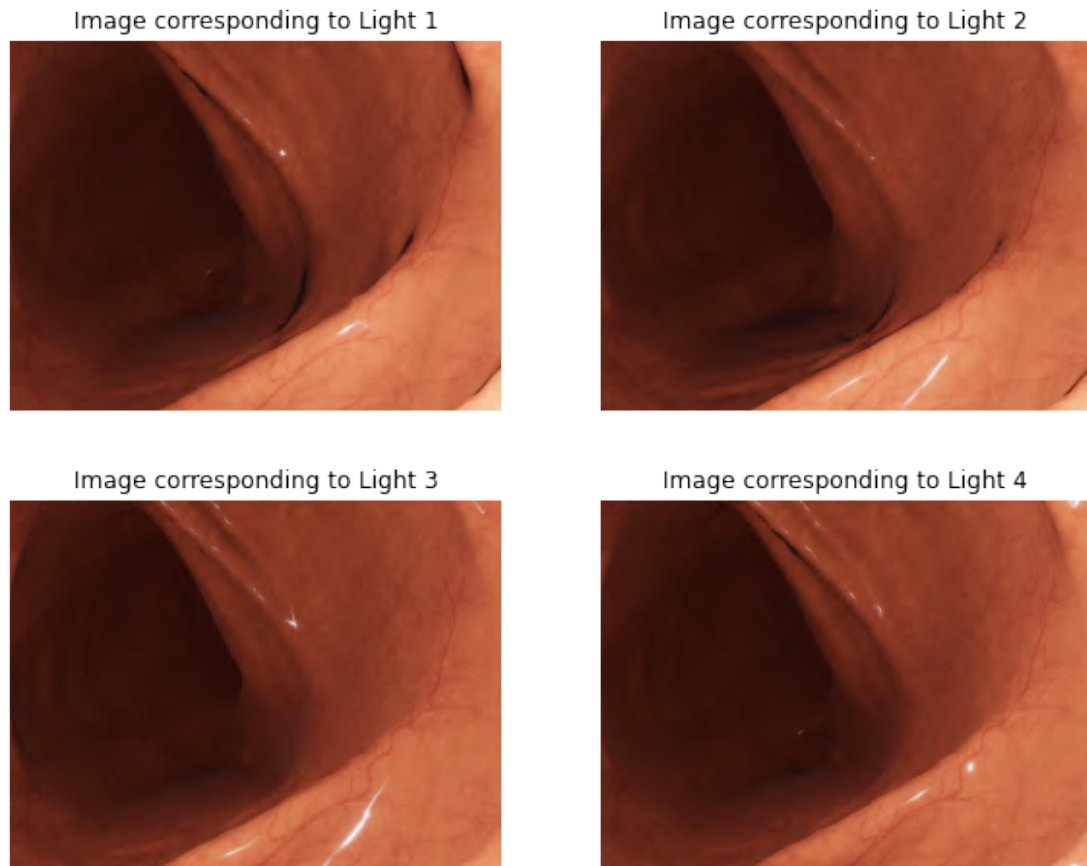


Figure 6.7: Images of an endoscopic scene (rendered in Unity)

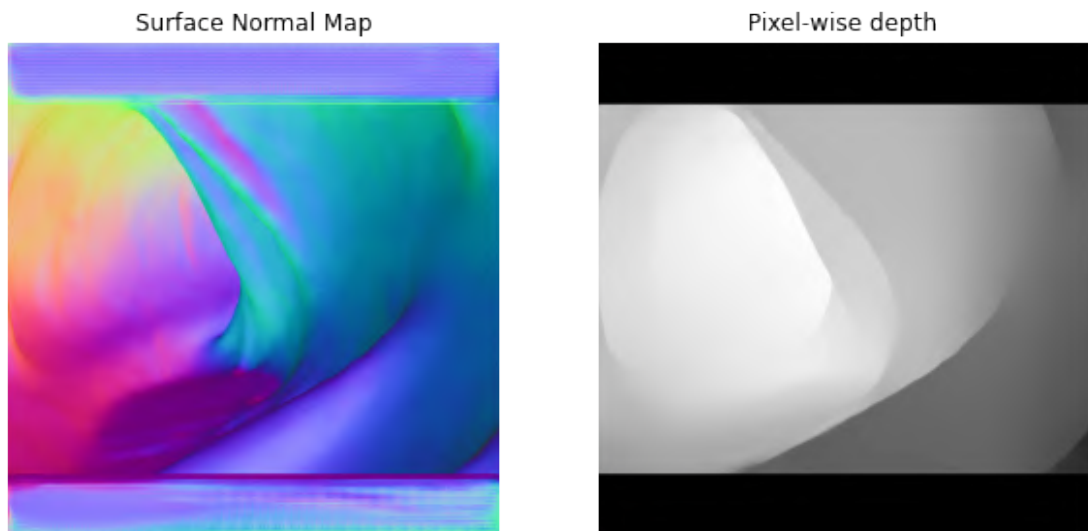


Figure 6.8: Predictions using the model trained on our rendered dataset

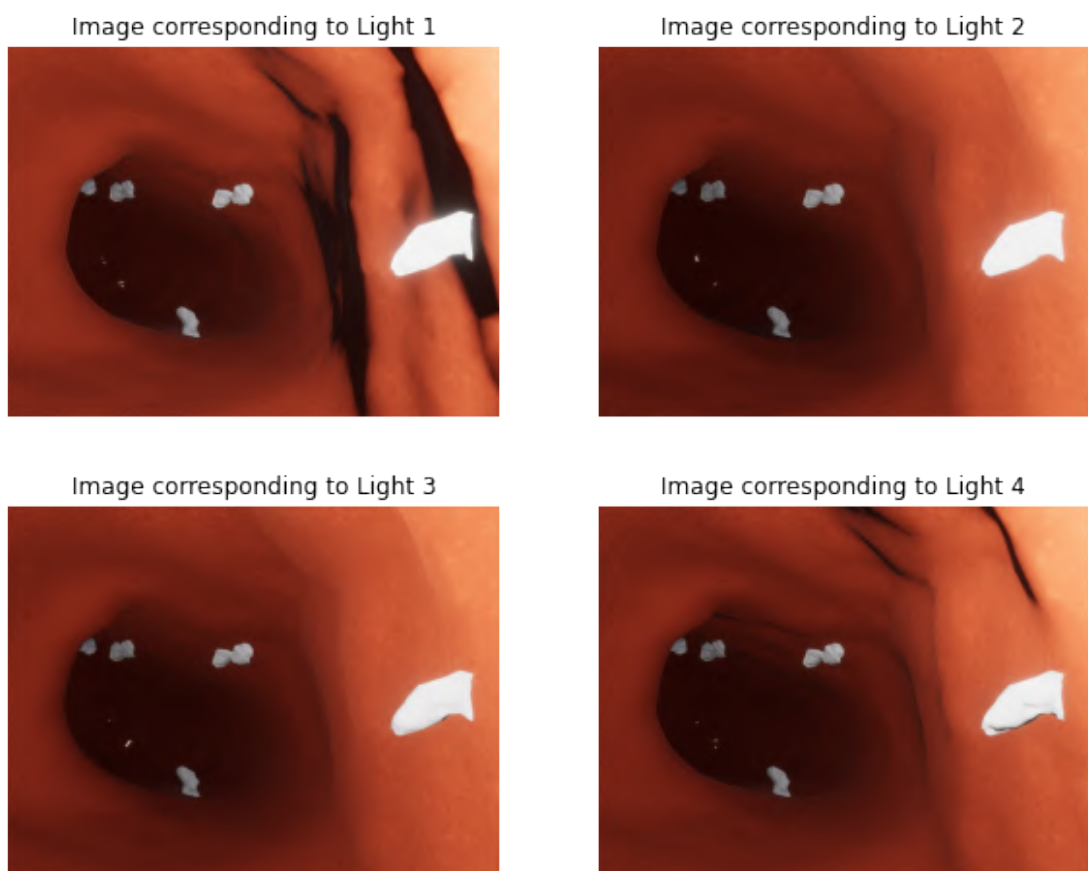


Figure 6.9: Images of another endoscopic scene (rendered in Unity)

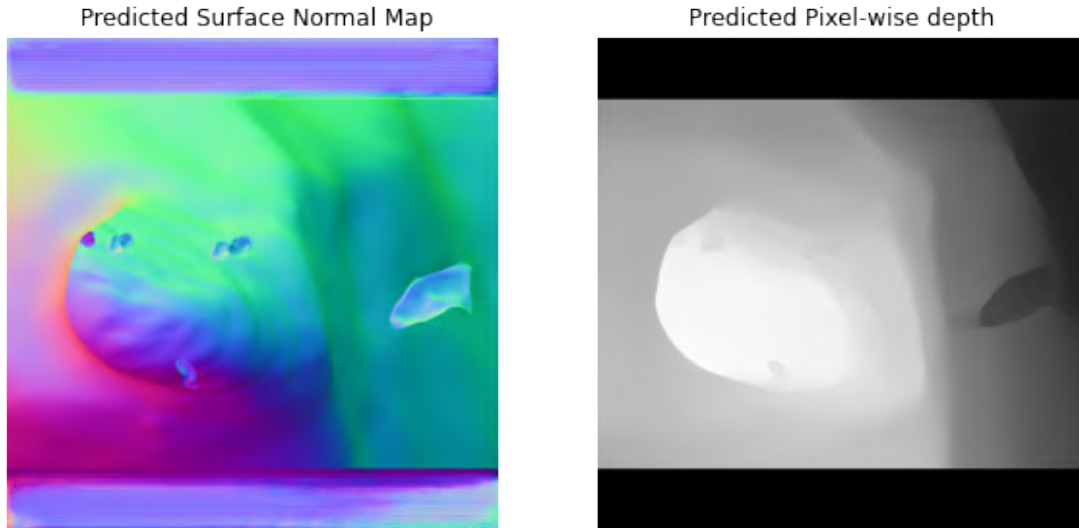


Figure 6.10: Predictions using the model trained on our rendered dataset

In Figure 6.11, on observing the training and validation loss curves for the total loss, mean angular error (i.e., mean of the angular error between the ground-truth and predicted surface normal vector) and the absolute depth error, we see that the training loss reduces in each epoch, but the validation loss curve plateaus and oscillates after a few epochs. This indicates that the model is overfitting the data. The problem of overfitting can be solved by rendering more endoscopic scenes and by using data augmentation techniques.

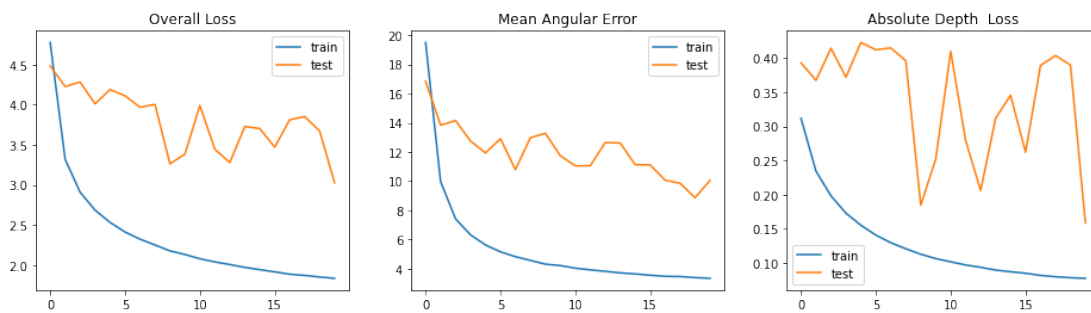


Figure 6.11: Training and Validation curves indicating overfitting

CHAPTER 7

Conclusion and Future work

In this thesis, we introduced the importance of developing an algorithm for obtaining the 3D information from endoscopic images. Knowledge of the 3D information, like pixel-wise depth and surface normals, helps in reconstructing three-dimensional structures of the endoscopic scenes, which help in improving the detectability and classification accuracy of cancerous tumours/lesions in the internal organs. The recovery of the three-dimensional structure of internal organs also helps in minimally invasive surgery. In this thesis, several methods based on Structure-from-Motion and Photometric stereo, both conventional and learning-based solutions, have been employed to obtain 3D information from endoscopic scenes.

The first method that we tested was COLMAP which is an improved version of the conventional Structure-from-Motion approach. We used traditional feature descriptors like SIFT and learning-based feature extraction (SuperPoint), and feature matching methods (SuperGlue). The results were not optimal because these networks were not trained on endoscopic data. Endoscopic images are smooth and textureless, so the number of distinct features detected is very low. We then used an end-to-end unsupervised learning approach for Structure-from-Motion, due to the lack of labelled ground-truth depth information for endoscopic images. The results were better than the conventional SfM method, but the problems that arise due to the smooth and textureless nature of endoscopic images persist.

We then shifted to photometric stereo, as it does not depend on correspondence matching and is capable of producing dense depth map estimates. In this thesis, the performance of conventional and near-field photometric stereo on images of ordinary objects and endoscopic images has been demonstrated. We decided to test out a learning-based approach for photometric stereo to improve the performance for endoscopic images. In order to train a deep learning model for photometric stereo, we created a synthetic photometric stereo dataset of endoscopic images using the Unity3D simulation environment. Using this dataset, we trained a model for near-field photometric stereo,

and the results are shown in this thesis. The model trained on this synthetic dataset generated by us will be used to predict the depth estimates of real-world endoscopic scenes that will be captured using our lab prototype of a photometric stereo endoscope (built using a Raspberry Pi camera and four LEDs) and some ex-vivo tissues.

The next step in this project is to collect more synthetic data using the pipeline explained in Chapter 5 and the Appendix. Generating more data will help overcome the problem of overfitting encountered while training our deep learning model for near-field photometric stereo. Another interesting direction to explore is the combination of information obtained from SfM-based methods and photometric stereo methods to improve the 3D reconstructions.

REFERENCES

- [1] Kutsev Bengisu Ozyoruk, Guliz Irem Gokceler, Taylor L. Bobrow, Gulfize Coskun, Kagan Incetan, Yasin Almalioglu, Faisal Mahmood, Eva Curto, Luis Perdigoto, Marina Oliveira, Hasan Sahin, Helder Araujo, Henrique Alexandrino, Nicholas J. Durr, Hunter B. Gilbert, and Mehmet Turan. Endoslam dataset and an unsupervised monocular visual odometry and depth estimation approach for endoscopic videos. *Medical Image Analysis*, 71:102058, 2021.
- [2] Vicente Parot, Daryl Lim, Germán González, Giovanni Traverso, Norman S. Nishioka, Benjamin J. Vakoc, and Nicholas J. Durr. Photometric stereo endoscopy. *Journal of biomedical optics*, 18(7):076017–076017, Jul 2013. 23864015[pmid].
- [3] Sang Bong Ahn, Dong Soo Han, Joong Ho Bae, Tae Jun Byun, Jong Pyo Kim, and Chang Soo Eun. The miss rate for colorectal adenoma determined by quality-adjusted, back-to-back colonoscopies. *Gut and liver*, 6(1):64–70, Jan 2012. 22375173[pmid].
- [4] Toby Collins and Adrien Bartoli. 3D reconstruction in laparoscopy with close-range photometric stereo. *Med Image Comput Comput Assist Interv*, 15(Pt 2):634–642, 2012.
- [5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 337–33712, 2018.
- [6] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4937–4946, 2020.
- [7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm

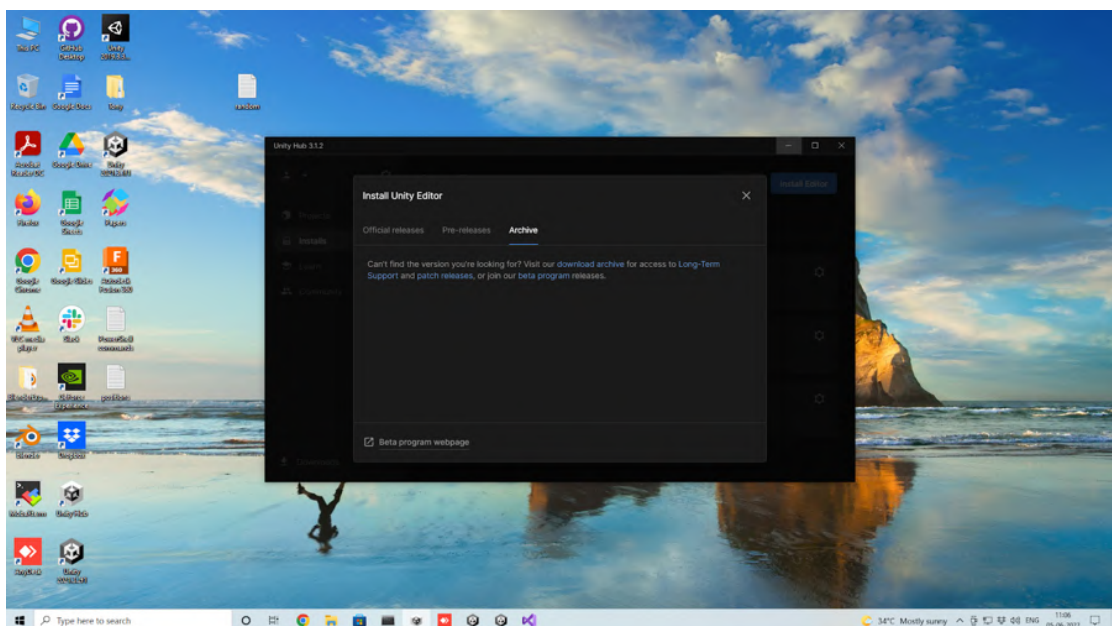
- for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981.
- [8] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.
 - [9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.
 - [10] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3827–3837, 2019.
 - [11] Edwin Olson, John Leonard, and Seth Teller. Fast iterative alignment of pose graphs with poor initial estimates. pages 2262 – 2269, 06 2006.
 - [12] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.
 - [13] Robert Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19, 01 1992.
 - [14] John K Haas. A history of the unity game engine. 2014.
 - [15] Kagan Incetan, Ibrahim Omer Celik, Abdulhamid Obeid, Guliz Irem Gokceler, Kutsev Bengisu Ozyoruk, Yasin Almalioglu, Richard J. Chen, Faisal Mahmood, Hunter Gilbert, Nicholas J. Durr, and Mehmet Turan. Vr-caps: A virtual environment for capsule endoscopy, 2020.
 - [16] Daniel Lichy, Soumyadip Sengupta, and David W. Jacobs. Fast light-weight near-field photometric stereo, 2022.
 - [17] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

CHAPTER 8

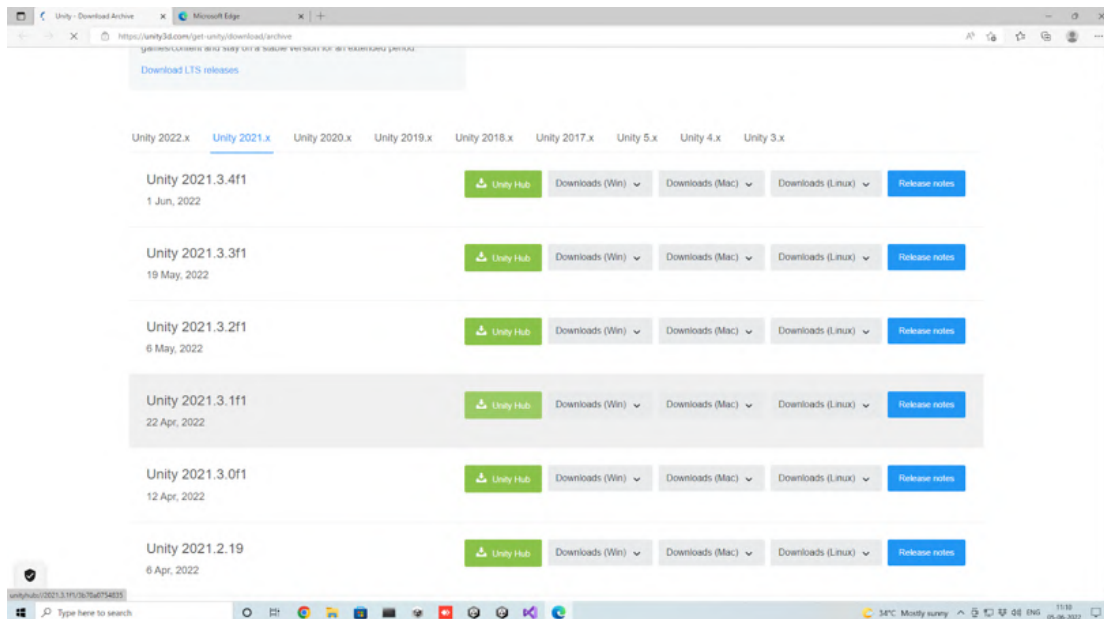
Appendix: Guide for Data Collection using Unity

8.1 Installation

1. Install UnityHub for Windows from <https://unity3d.com/get-unity/download>.
2. Follow the installation wizard and create a Unity account when prompted to do so.
3. Open UnityHub and go to the “Installs” tab. Click on the “Install Editor” button. To get a specific version of Unity (if it is not listed in the Official releases tab), switch to the Archive tab and follow the instructions.



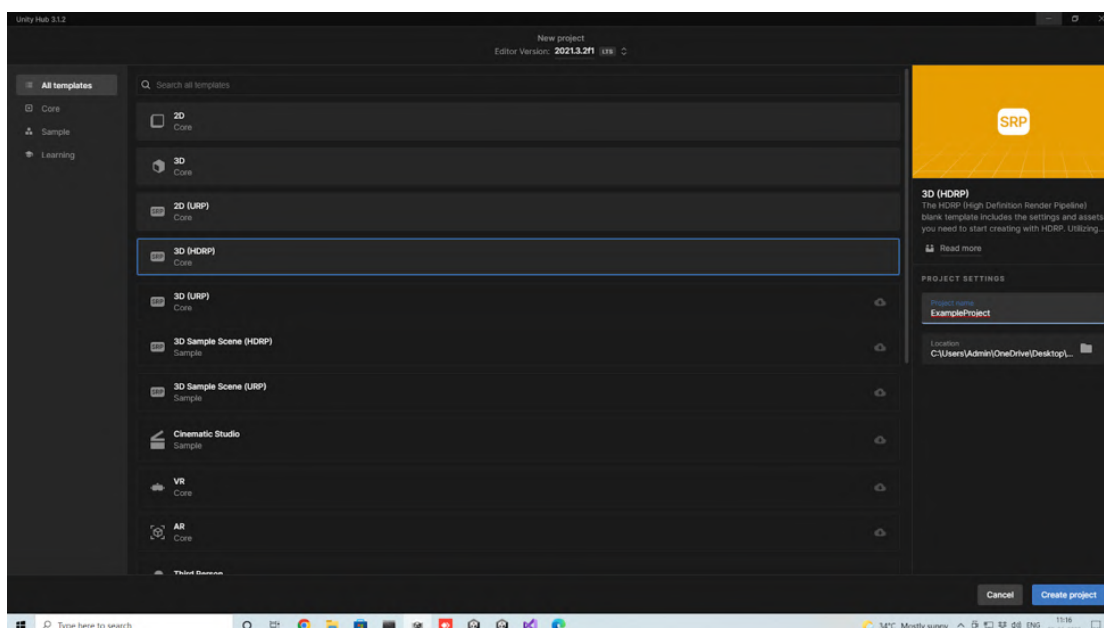
4. Find the version of Unity you need from the archive and download it using UnityHub, like so:



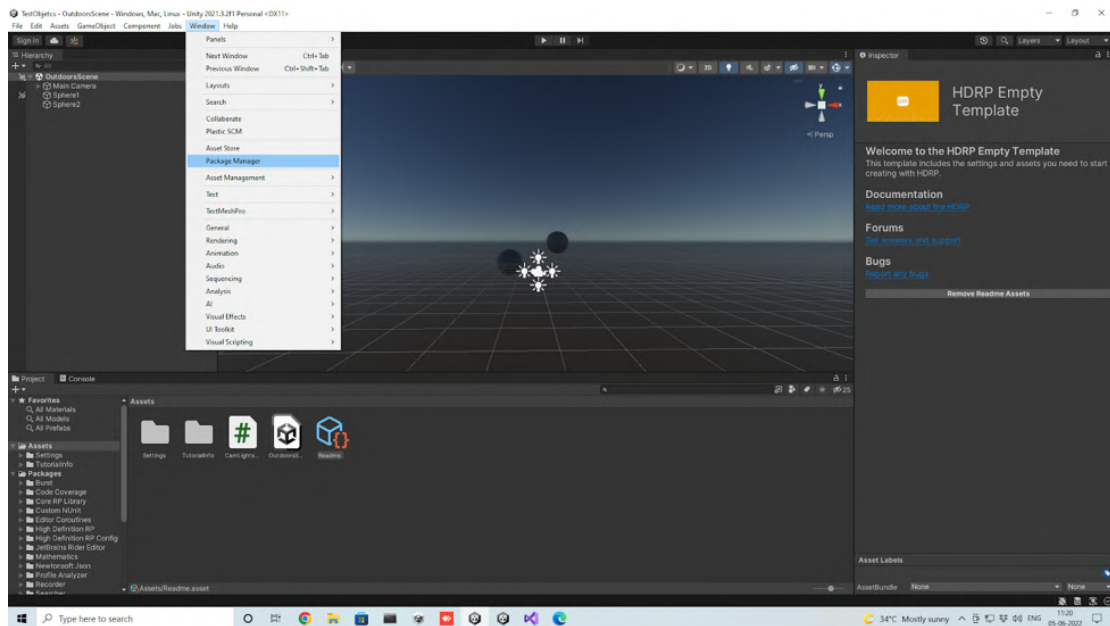
5. The version of Unity that I have used is 2021.3.1f1. The version used in the VR Caps [15] project is 2019.3.3f1. Download both versions. It is possible to have multiple versions of Unity installed on your computer. Remember to select the correct version while creating a new project.

8.2 Creating a new project and downloading the recorder:

1. Go to the Projects tab and click on the new project button. The following screen opens. Choose a version of Unity, set the project name and location and click “Create Project”.



2. You have to use the 3D HDRP template to create a new project which is capable of using the Unity AOV Recorder for recording the depth and surface normals of the scene.
3. Open the project. Note that when you create a new project or import a project for the first time, there will be a significant delay. Click on the “Window” menu and select Package Manager.



4. Select the “Unity Registry” option and find the “Recorder” package and click “Install” in the right pane.

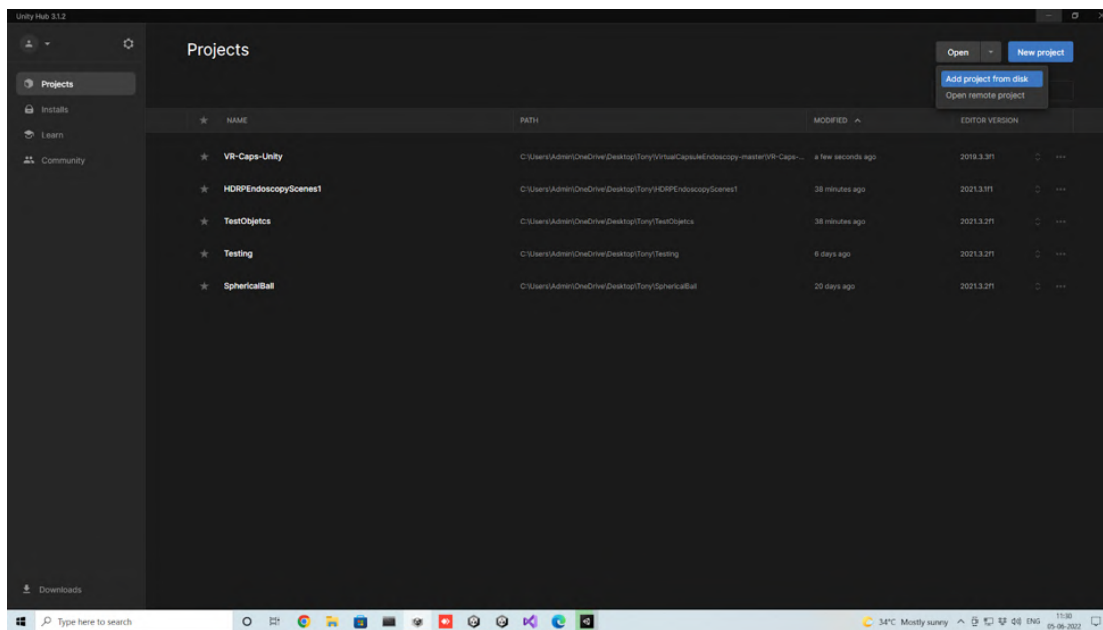
Unity Image Recorder manual

Unity AOV Recorder manual

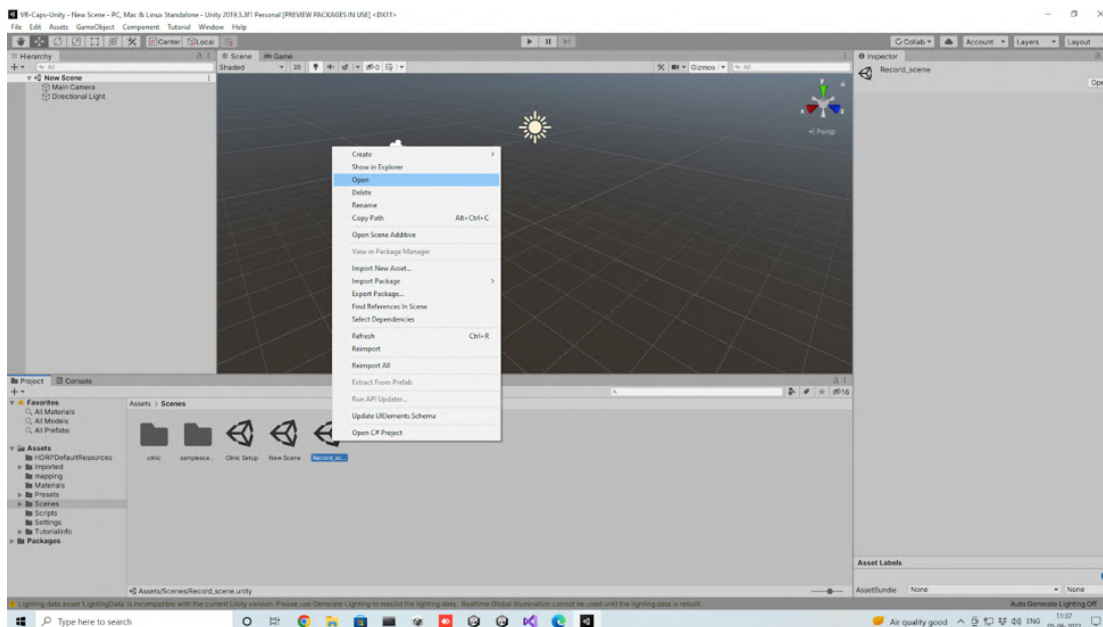
Please note that the recorder stores the images in sRGB format and each pixel intensity value has to be raised to 2.2 to get RGB values.

8.3 Importing the organ models:

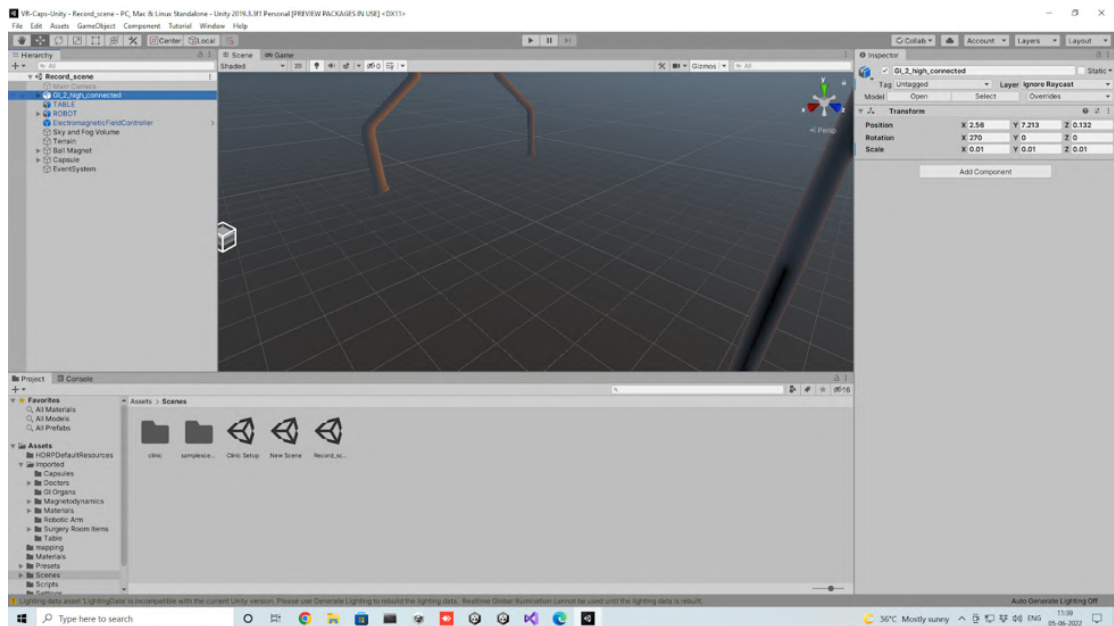
1. Open the VR-Caps-Unity project with the correct version of Unity (2019.3.3f1). This project can be downloaded from here.
2. Add this project to UnityHub. Go to the Projects tab and click on the arrow next to “Open” at the top right corner and select “Add project from disk”. Find the project in the location where you downloaded VR-Caps-Unity and click “Add Project”.
3. Open the project by double-clicking on it in Unity Hub once it is added.



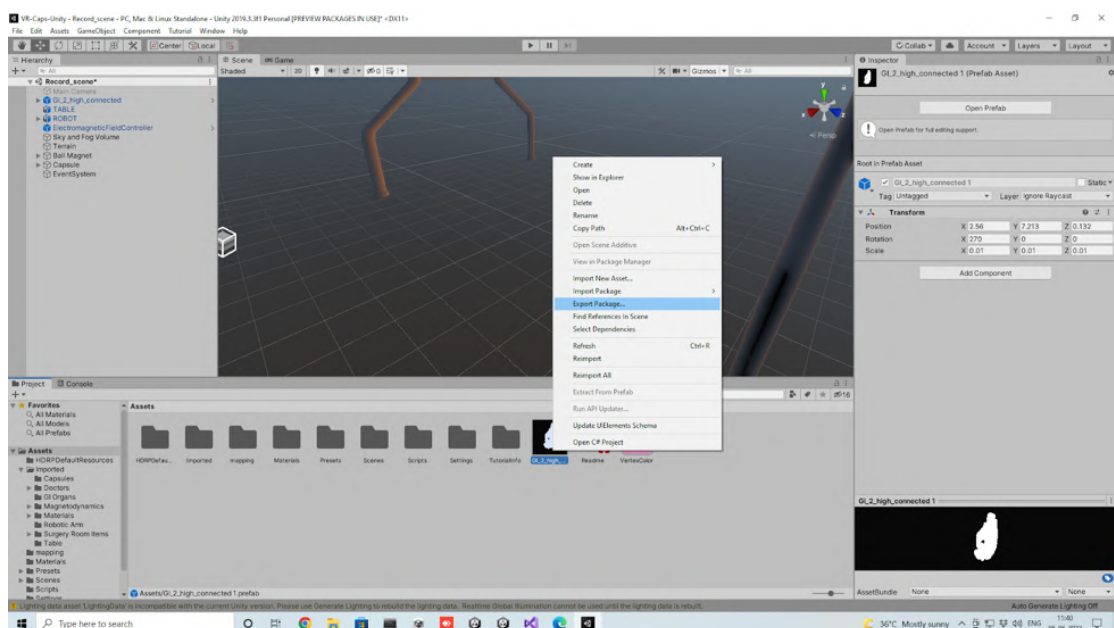
4. Open the “Record_scene”:

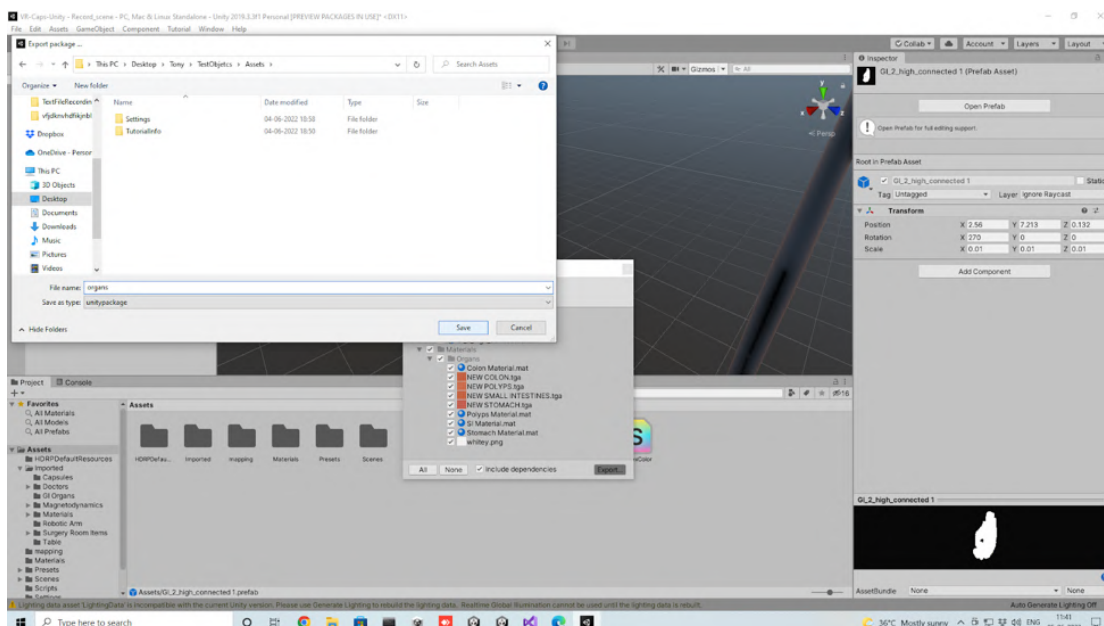
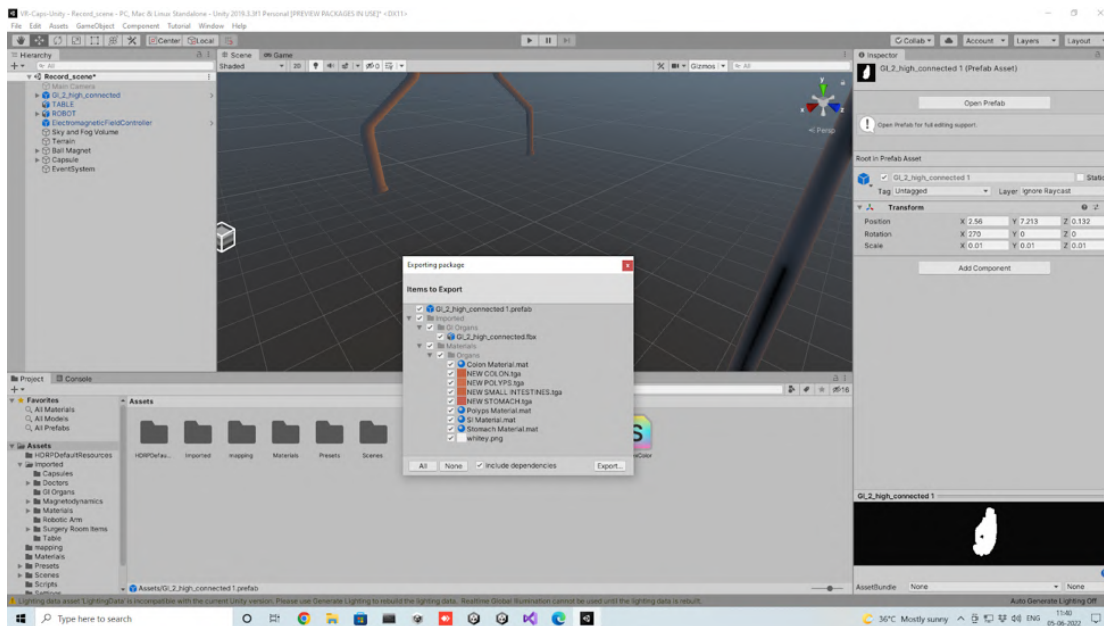


5. In Record_scene, drag the GI_2_high_connected into the Assets section at the bottom left to create a new Prefab.

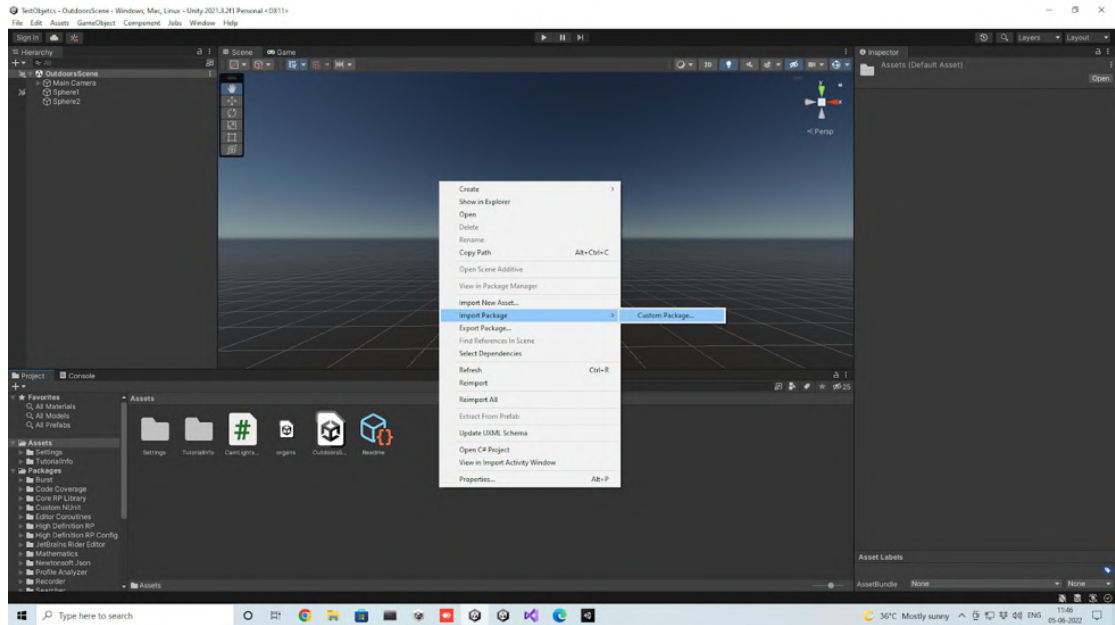


- Right-click on the newly created Prefab and select Export Package. Save it in the file format of “unitypackage” in the Assets folder of the project that you want to import these organs into.

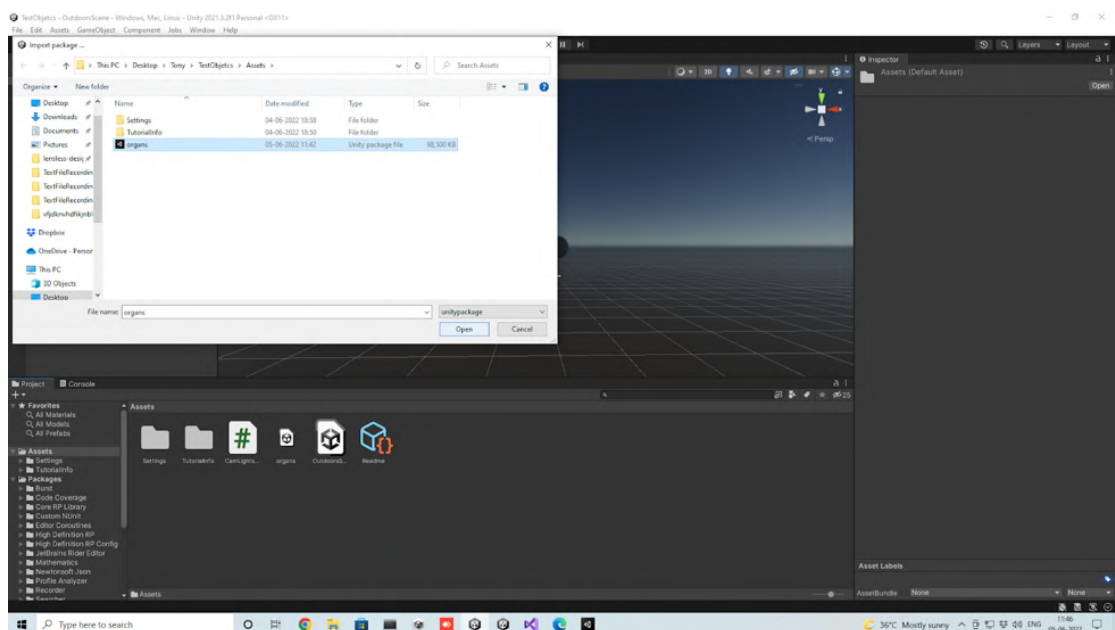


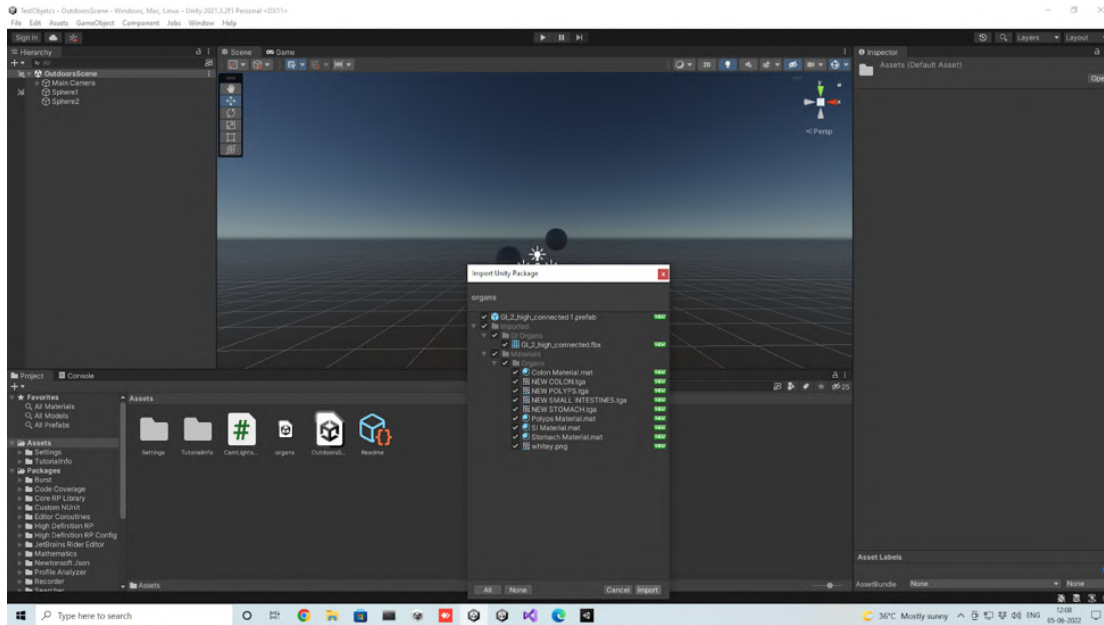


- Open the project where you want to import the organs. Right-click anywhere in the empty space after selecting “Assets” in the Project window (lower half of the screen). Click Import Package->Custom Package.

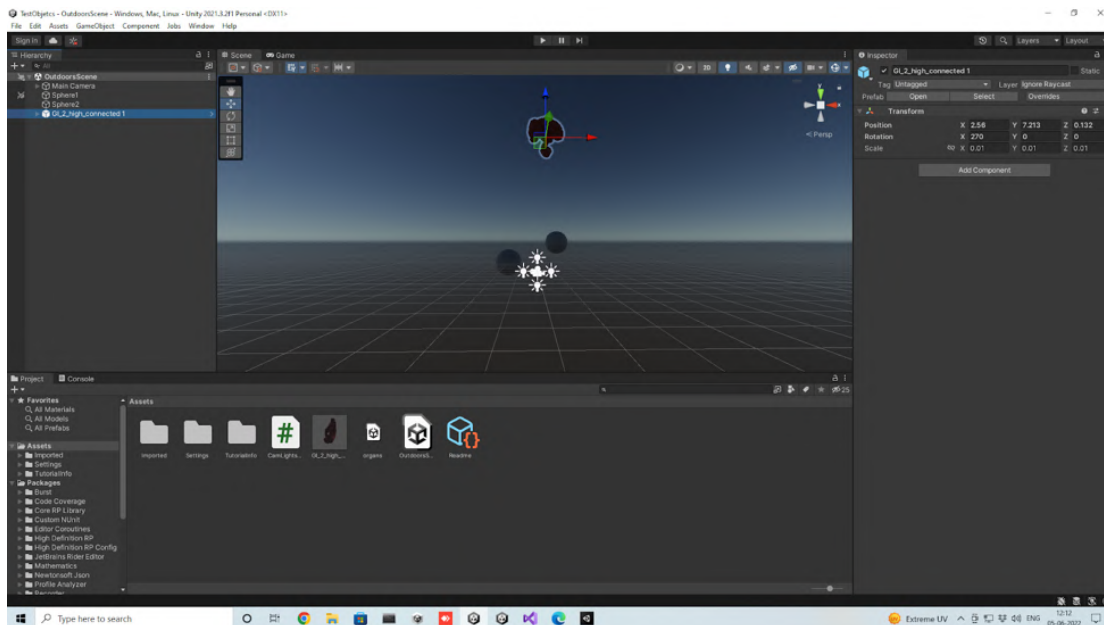


8. Find the Prefab that we created in the previous steps and select “Open” to import the Prefab into the new project.





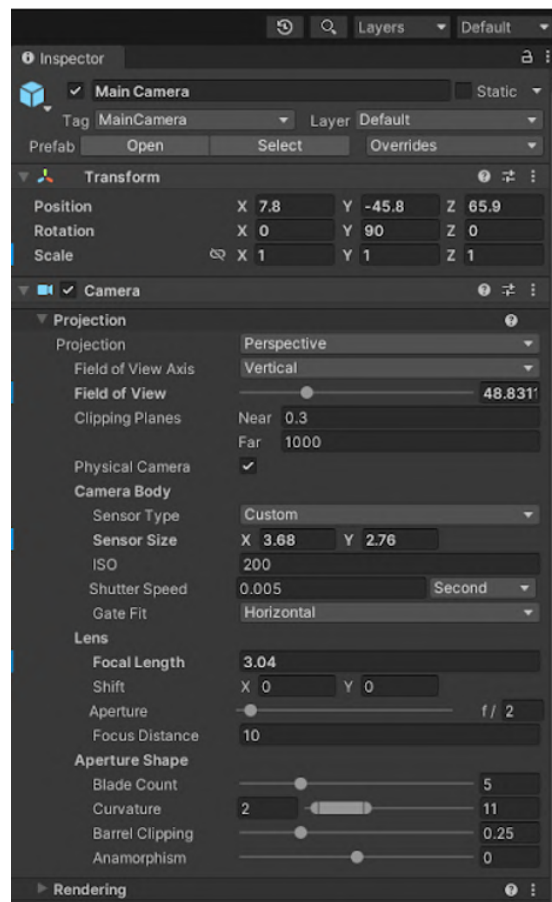
9. Drag the package from the Assets window into the Scene Hierarchy window. The organs get added to the scene.



8.4 Camera Parameters

To set camera parameters like focal length, sensor size etc. first check the “Physical Camera” checkbox. The focal length and the sensor size in X and Y directions have to be set in millimetres. If these values are set correctly, then Unity automati-

cally calculates the field of view angle. Ensure that the value matches with the camera specifications. You can also set the aperture value if you know the F-stop of the camera.



You can find the specs for the Raspberry Pi camera here. This is the camera that we are trying to mimic in Unity.

8.5 Data Collection using Unity

The camera parameters are set as described above. There are four light surrounding the camera. Each light is created as a Point Light with range 10 and when it is switched on, the intensity value is set at 600000 Lumen. The camera and light setup is created to mimic this setup:

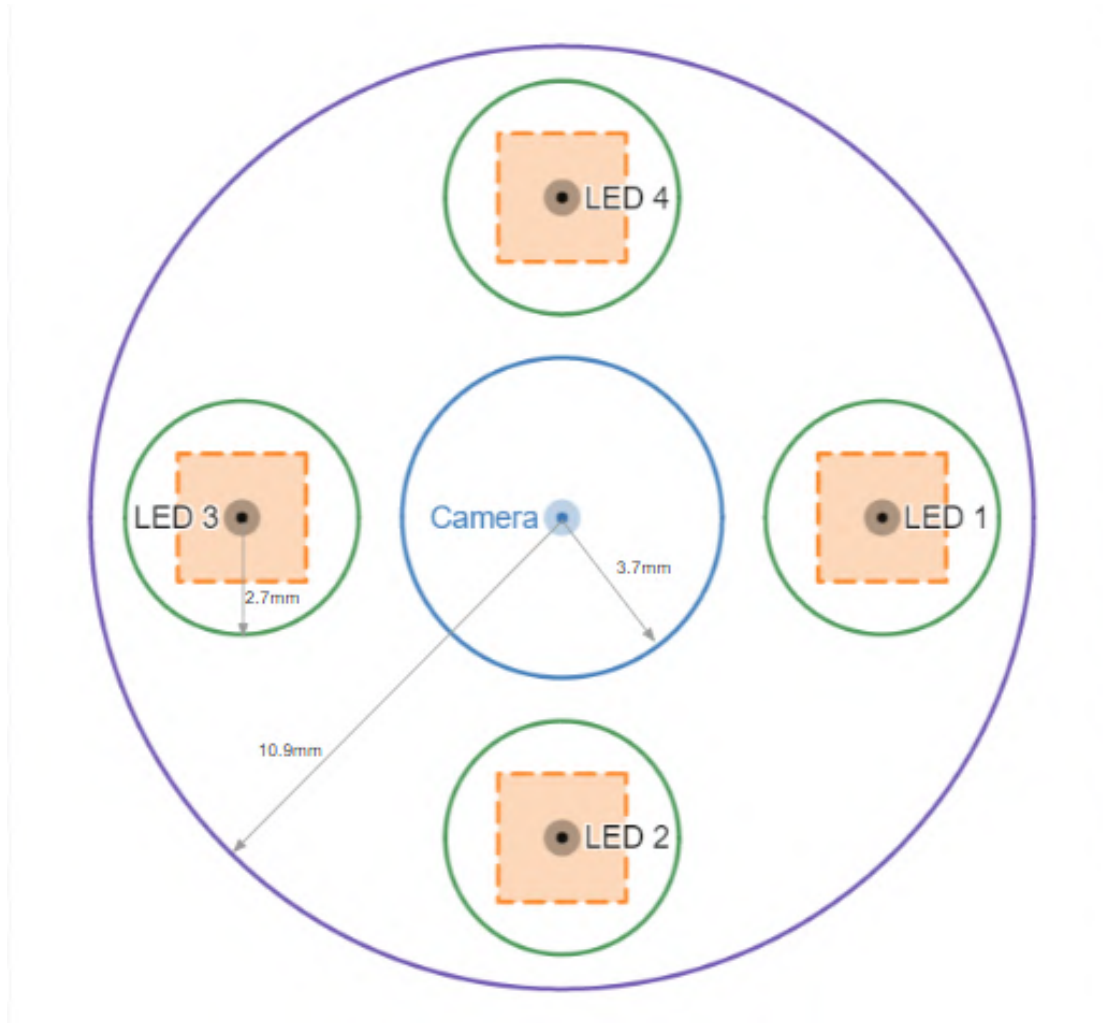


Figure 8.1: Diagrammatic representation of the Camera + LEDs model

The big circle in the centre is for the camera. The other 4 smaller circles are for LEDs and the square in each circle is the region from which the LED positions are randomly sampled. The random sampling is done to make the neural network robust to errors in the light positions in the real 3D-printed model.

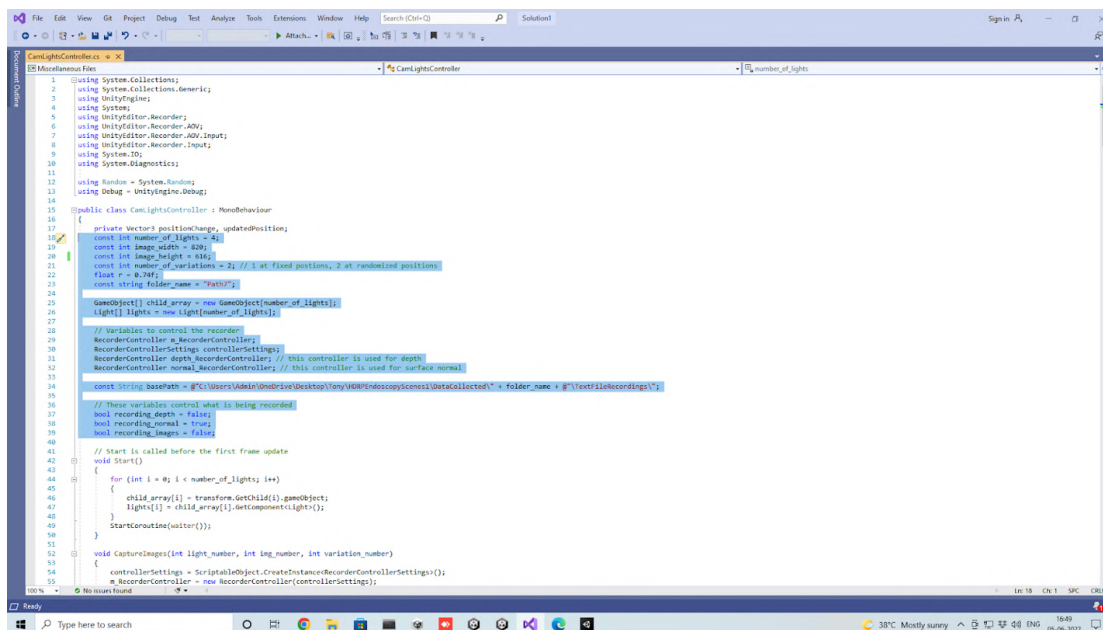
The translation and rotation of the camera, switching on and off of the LEDs are controlled by the code. At each position of the camera, the orientation of the camera is changed 19 times. At each rotation, first we keep the LEDs at their fixed positions, then the positions are randomly sampled from within the orange square region shown in the image. Since we have one fixed set of positions, and the sampling is done twice, at each rotation, we get 12 images of the scene. The Unity Image Sequence Recorder stores the image, and also the location and orientation of the camera, LEDs, both absolute and relative to the camera. The Unity AOV Recorder stores the depth and the surface normals.

8.6 Steps for Data Collection

The code that controls the camera and lights is in the file `CamLightsController.cs`. To add such a script in a new project, select the camera (Main Camera) in the Inspector pane, click on “Add Component”. Select “New script”, enter the file name, and click “Create and Add”. A script is added to the camera object. It can be opened by clicking on the script in the Assets window.

1. Create a folder named “Data Collected” in the main project folder. Inside this folder, create a folder named `any_folder_name`. Inside the folder, create another empty folder named “TextFileRecordings”. The information about the camera and LED location, orientation, absolute depth of the centre pixel etc. gets stored in this folder.
2. The images, depth and surface normals will be saved in the “ImageRecordings”, “DepthRecordings”, “SurfaceNormalRecordings” folders respectively. These folders are all created automatically when the code runs.
3. File naming convention:
 - Images: `image_<p>_variation_<q>_light_<r>.png` represents an image collected at the p^{th} position with the r^{th} light switched on and all others switched off. If q is 0, it means that the LEDs are at their fixed positions. If q is greater than 0, the q is the variation number. `image_<p>_noLights.png` represents the image collected with all lights switched off.
 - Depth: `depth_image_<p>.png` represents the depth image collected at the p^{th} position.
 - Surface Normal: `surface_normal_image_<p>.png` represents the surface normal image collected at the p^{th} position.
4. The `image_width` and `image_height` variables control the shape of the images and depth and surface normal maps that the Recorder stores.
5. The variable `number_of_variations` stores how many times LED positions need to be sampled from the square regions.
6. The variable `folder_name` must be the name of the folder inside “DataCollected”, it corresponds to “`any_folder_name`” mentioned in step 1.
7. The floating point variable r is the radius of the circle on which the centres of the LEDs lie. If the size of the 3D printed model changes, then this variable will need to be changed. The current value is 0.74cm.
8. The variable `basePath` has to be modified based on the location of the project.
9. There are three boolean variables `recording_depth`, `recording_images`, `recording_normals`. Only one of these must be true and the other two must be false at any time. If `recording_images` is true, then the images are recorded and stored

in the ImageRecordings folder. The text files are also populated with camera location, orientation, LED position, absolute depth of the centre pixel etc.



```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using System;
5 using UnityEditor.Recorder;
6 using UnityEditor.Recorder.AOV;
7 using UnityEditor.Recorder.AOV.Input;
8 using UnityEditor.Recorder.Input;
9 using System.IO;
10 using System.Diagnostics;
11
12 using Random = System.Random;
13 using Debug = UnityEngine.Debug;
14
15 public class CamLightController : MonoBehaviour
16 {
17     private Vector3 positionChange, updatedPosition;
18     const int number_of_lights = 4;
19     const int image_width = 820;
20     const int image_height = 616;
21     const int number_of_variations = 3; // 1 at fixed positions, 2 at randomized positions
22     float r = 0.785f;
23     const string folder_name = "Path00";
24
25     GameObject[] child_array = new GameObject[number_of_lights];
26     Light[] lights = new Light[number_of_lights];
27
28     // variables to control the recorder
29     RecorderController m_RecorderController;
30     RecorderControllerSettings controllerSettings;
31     RecorderController depth_RecorderController; // this controller is used for depth
32     RecorderController normal_RecorderController; // this controller is used for surface normal
33
34     const string basePath = @"C:\Users\Udwin\OneDrive\Desktop\Unity\HMD\Indocopy\scenes\UdataCollected\" + folder_name + @"\" + "extrileRecordings\";
35
36     // These variables control what is being recorded
37     bool recording_depth = false;
38     bool recording_normal = true;
39     bool recording_images = false;
40
41     // Start is called before the first frame update
42     void Start()
43     {
44         for (int i = 0; i < number_of_lights; i++)
45         {
46             child_array[i] = transform.GetChild(i).gameObject;
47             lights[i] = child_array[i].GetComponent<Light>();
48         }
49         StartCoroutine(waiter());
50     }
51
52     void CaptureImages(int light_number, int img_number, int variation_number)
53     {
54         controllerSettings = ScriptableObject.CreateInstance<RecorderControllerSettings>();
55         m_RecorderController = new RecorderController(controllerSettings);
56     }
57 }

```

10. positionChange defines how the camera will translate along X, Y, Z axes. It is in the third line of the IEnumerator waiter function.

11. The floating point arrays rotation_angles_x, rotation_angles_y, rotation_angles_z store the angle by which the camera will be rotated along that axis. The rotation will be such that we pick a value from one of these arrays and the rotation along the other two axes will be zero.

8.7 Text File data

Recordings

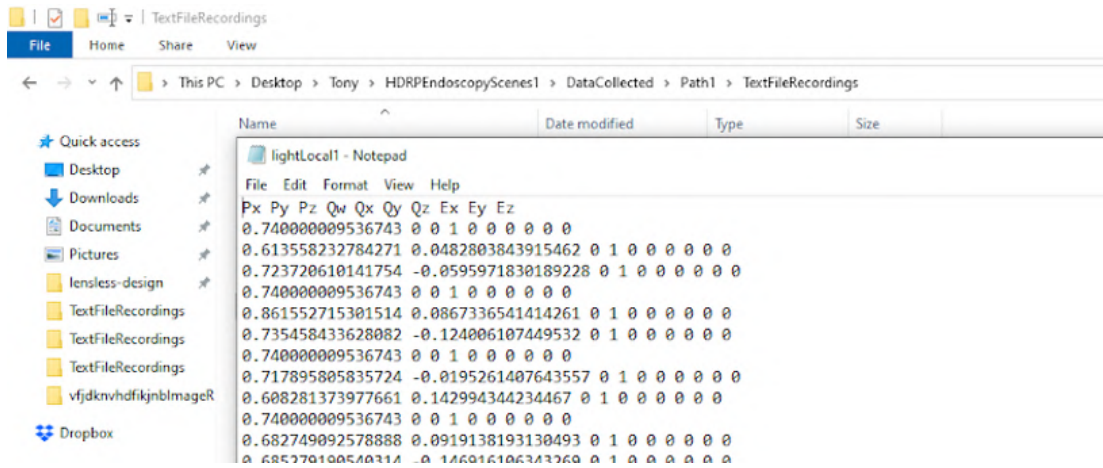
View

PC > Desktop > Tony > HDRPEndoscopyScenes1 > DataCollected > Path1 > TextFileRecordings

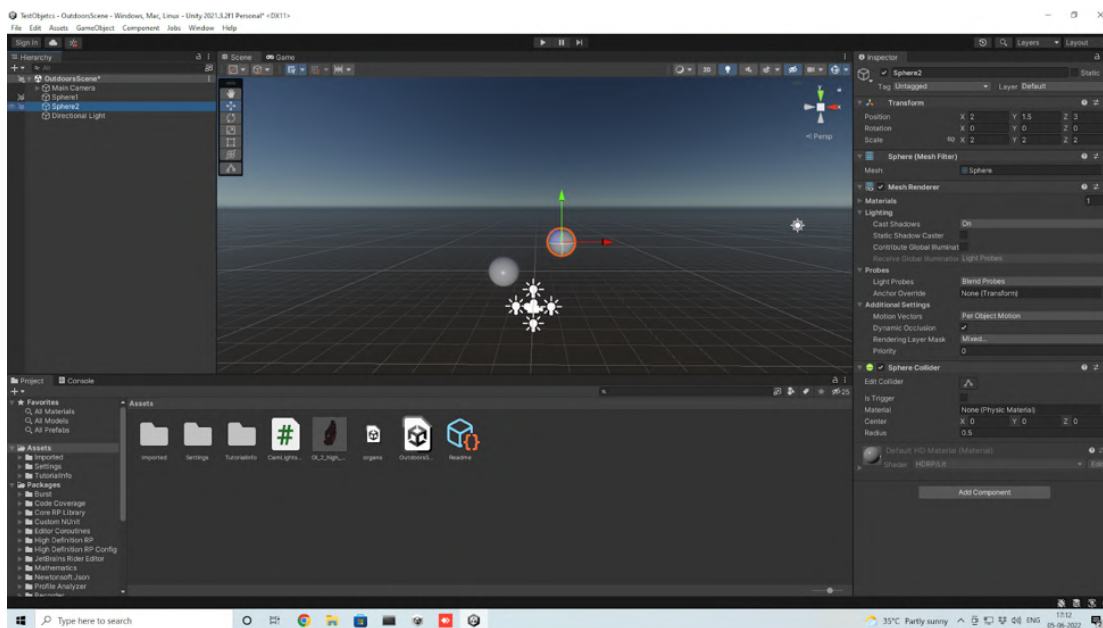
	Name	Date modified	Type	Size
	camera	27-05-2022 20:36	Text Document	29 KB
	centre_pixel_absolute_depth	27-05-2022 20:36	Text Document	2 KB
	light1	27-05-2022 20:37	Text Document	101 KB
	light2	27-05-2022 20:37	Text Document	101 KB
	light3	27-05-2022 20:37	Text Document	101 KB
	light4	27-05-2022 20:37	Text Document	100 KB
	lightLocal1	27-05-2022 20:37	Text Document	27 KB
	lightLocal2	27-05-2022 20:37	Text Document	28 KB
	lightLocal3	27-05-2022 20:37	Text Document	28 KB
	lightLocal4	27-05-2022 20:37	Text Document	27 KB
ieR	time_taken	27-05-2022 21:25	Text Document	1 KB

- camera.txt: Each line stores the XYZ coordinates of the camera, followed by the orientation of the camera in quaternion format, followed by the orientation of the camera in Euler angles (degrees).
- centre_pixel_absolute_depth.txt: On each line, we have the absolute depth of the centre pixel.
- lightLocal<x>.txt: It stores the XYZ coordinates of the LED, followed by the orientation of the LED in quaternion format, followed by the orientation of the LED in Euler angles (degrees). Note that everything in this file is with respect to the camera as the LEDs are child objects of the Main Camera.

The first line below the header shows the values for LED 1. The next two lines are also for LED1, but these correspond to the cases where the position of the LED is sampled from a region around the correct position. The fourth line shows the fixed LED positions again, as it corresponds to the next orientation of the camera. Every three lines (1 fixed + 2 number_of_variations) correspond to one position/orientation of the camera.



8.8 Calibration setup



- Adjust the XYZ coordinates and the rotation along the axes in the “Transform” section of the Inspector window.
- The SphereCollider radius is 0.5. The radius of the sphere is the product of the Scale and the SphereCollider radius.
- Directional Light (Sun) must be removed from the scene before doing any data collection. Remember that for directional light, its location does not matter, only the rotation matters.