Surgical Instrument Tracking for Intraoperative Vitrectomy Guidance Using Deep Learning and Stereo Vision

BY

MATTIA DI FATTA
B.S., Politecnico di Milano, Milan, Italy, 2017

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:

Tanya Berger-Wolf, Chair and Advisor
Cristian Luciano, Department of Bioengineering
Yannek Leiderman, Department of Ophthalmology
Wei Tang
Marco Domenico Santambrogio, Politecnico di Milano

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Cristian Luciano, and Dr. Yannek Leiderman for the opportunity and their precious and continuous help that made my work much easier during the last year. I deeply thank Professor Tanya Berger-Wolf for her valuable technical advice and for the numerous soft-skills tips she taught me. I am grateful to all the colleagues and friends of the Mixed Reality Lab who always showed their support. A deep thank you to all the fantastic people I knew in Chicago, to all my fellow Italians who started this adventure with me and in particular to Flavio and Davide, my roommates. A special thank you to Gabriele with whom I spent unforgettable moments with and who helped me so many times during these months.

A thank you to Marco, Cinzia, and Sergio for your continuous and deep support even from the other side of the world.

Finally, a special thank you to my parents who allowed me to start this marvellous experience and who supported me every single day, although so far from home.

MDF

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHE	Adaptive Histogram Equalization
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
BRIEF	Binary Robust Independent Elementary Features
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma Separated Values
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
FAST	Features from Accelerated Segment Test
FFNN	Feed-Forward Neural Network
FHD	Full High-Definition
FPS	Frame Per Second

LIST OF ABBREVIATIONS (continued)

GPU	Graphical Processing Unit
GUI	Graphical User Interface
HDD	Hard Disk Drive
IDE	Integrated Development Environment
IO	Input Output
iOCT	Intraoperative Optical Coherence Tomography
MDP	Markov Decision Process
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
OCT	Optical Coherence Tomography
ORB	Oriented FAST and Rotated BRIEF
RAM	Random Access Memory
RGB	Red Green Blue
RNN	Recurrent Neural Network
ROI	Region Of Interest
SGBM	Semi-Global Block Matching
SIFT	Scale Invariant Feature Transform
SSD	Solid State Drive

SUMMARY

Surgeries are always challenging procedures. In ophthalmology in particular, the main difficulties are represented by limited space, complicated viewpoints, and bad light conditions, which make it even harder for surgeons to operate safely. This means avoid touching the retina, i.e. the back surface of the eye, perform secure movements and do not apply too much force on surfaces. This thesis deals with the problem of retinal collision avoidance through a proposed real-time pipeline utilizing both Deep Learning and Computer Vision to provide ophthalmologists with additional information about depth perception.

CHAPTER 1

INTRODUCTION AND MOTIVATION

Vitrectomy is an eye surgery during which ophthalmologists remove the vitreous *i.e.*, the gel filling of the eye cavity, to allow better access to the retina, the back surface of the eye. This procedure enables to perform various kinds of retina repairs, among which are scar tissue removal, retina detachments repair, and treatment of macular holes.

All the mentioned eye surgeries require ophthalmologists to operate with instruments in the range of 10 mm from the retinal surface. This distance is critical as the retina is an extremely sensitive and frail surface that may be seriously damaged by any quick movement or high pressure applied to it. Any damage to the retina can provoke serious eye conditions, among which the worst case is surely blindness. For these reasons, safety is a huge concern in ophthalmology and any viable tool that can, even slightly, enhance safety is adopted.

In these scenarios, safety can be enforced by:

1. ensuring a safety distance between instruments and the retina's surface;
2. limiting the force applied to the retina;
3. limiting the speed of movements.

With this work, we focus on the first point only.

Currently, the techniques ophthalmologists can rely on to ensure the safety distance aforementioned consist of simple, visual cues and experience of the surgeon. However, these approaches are highly unreliable due to a variety of factors.

1.1 Thesis goal

For what has been said in Chapter 1, the goal of this work is to provide additional, reliable information about the distance of the instrument from the retina that the surgeons can couple with visual cues to avoid retinal collisions and help decrease the risk of damages to patients. Besides, to make this information available to ophthalmologists during surgeries, this work also has the goal of providing the necessary information in real-time.

1.2 Outline

The rest of the thesis is organized as follows. Chapter 2 provides quick background knowledge of the structure of the eye bulb together with an overview of adopted and proposed solutions to previously mentioned limitations in ophthalmology to contextualize the work done in this thesis. Chapter 3 then discusses related works employing stereo vision and deep learning in various fields among which ophthalmology and some works with the similar goal of this thesis. In Chapter 4, we illustrate the theoretical aspects and terminology of the algorithms used and formally state the problem we want to face. Chapter 5 takes the reader through the specific adaptation of the methodologies presented in Chapter 4 stating the reasons for the adoption. Continuing in Chapter 6, the reader finds the technical description of the data used and the specification of the low-level details of the implemented solution, including parameters' values and programming environments. Chapter 7, then, presents a discussion of the obtained results.

Finally, Chapter 8 recaps all the work done and draws the conclusions and possible future work starting from the achieved results.

CHAPTER 2

BACKGROUND

2.1 Human eye structure

Given the purpose of this work and its area of application, we now present an overview of the structure of a human eye. This overview does not pretend to be comprehensive of all the medical details involving the human eye but aims at providing the knowledge required for this dissertation.

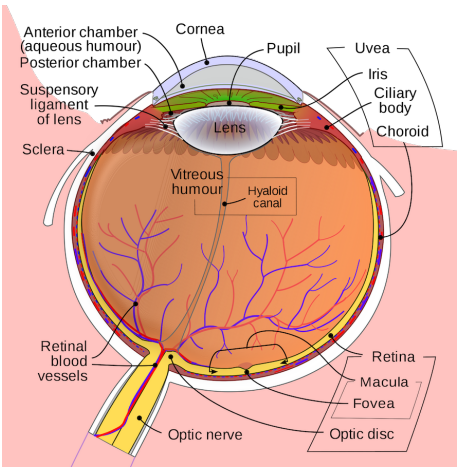

Figure 1: Human eye structure.*

*By Rhcastilhos. And Jmarchn. - CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1597930

As shown in Figure 1, the human eye is a sphere-like structure that can be roughly divided into an anterior part (or segment) whose main components are the cornea, the iris, and the pupil, and a posterior part, which comprehend many more sub-components [1]. The surface of the eye bulb's sphere is composed of three layers. From the outer to the innermost they are:

- the fibrous tunic, composed of the cornea on the front and the sclera on the back;
- the uvea, consisting of the choroid, the ciliary body, and the iris on the front;
- the retina, which presents retinal and blood vessels.

Light enters in the eye through the cornea, the pupil and then through the lens, whose shape can be changed to adjust the focus. The size of the pupil controls the amount of light entering the eye. Once inside the eye, photons are captured and converted to electrical signals by the photo-sensitive cells of the retina and transmitted to the brain through the optic disc. The posterior portion of the eye, between the retina and the lens, is filled with the vitreous, a substance made of water and proteins with sticky density.

2.2 Vitrectomy

There exist many different eye diseases, many of them concerning the posterior part of the eye. As can be read in section 2.1 and seen in Figure 1, many components residing on the back of the eye are not easily accessible if affected by a disease. In particular, the retina, which can be affected by many conditions, is complex to be accessed.

Vitrectomy [2] is the name of the ophthalmic surgery, which is performed to enable easy access to the retina by removing the vitreous. There exist two main types of vitrectomy. The

first is called *Pars Plana Vitrectomy* and is performed to allow further operations for diseases affecting the posterior segment of the eye. The second type of vitrectomy is called *Anterior Vitrectomy* and, as can be deducted by the name, is performed in case of conditions affecting the anterior portion of the eye, in particular, the anterior chamber. Rarely, in fact, the vitreous can flood in the space between the cornea and the pupil, requiring surgery. After the vitrectomy is complete, the posterior segment is filled with a gas bubble that is used to keep the retina in place until the eye heals completely.

2.3 Surgical Microscopes and TrueVision System

To successfully perform eye surgery like vitrectomy, the ophthalmologist should have a clear view of the inside of the eye. To enable this clear view, surgical microscopes [3] are currently employed in the operating room to provide a magnified view of both the anterior and the posterior segment of the eye. However, this magnification of the eye would be useless without the employment of a correction lens placed by ophthalmologists between the microscope's lenses and the eye to take into account distortion produced by the cornea, the eye lens and the humor. Besides this assistantship, these surgical microscopes provide more advanced visual assistance, such as 3D visualization and Optical Coherence Tomography that we describe in more details in the following sections. Since surgeons have to look through the surgical microscope, they lose any perception of depth as they look at a 2D image. 3D cameras setups, such as TrueVision 3D System by Leica Microsystems [4], can be attached to stereoscopic microscopes to help ophthalmologists on this aspect. These systems display and record a stereo video of the surgery in real-time.

2.4 Visual Cues

The use of the surgical microscope described in section 2.3 enables a variety of simple visual cues. This approach may seem trivial but visual cues are still a powerful tool in ophthalmology widely used by ophthalmologists to operate safely [3]. Probably, the simplest and most widely used visual clue is the distance between the shadow of the instrument produced by a surgical torch and the instrument's tip. Intuitively, the closer the shadow to the instrument's tip in the 2D view, the closer the instrument's tip to the retina in the 3D space. However, this approach is extremely unreliable due to the complex light conditions in the eyeball caused by the limited space where ophthalmologists have to operate. Moreover, the surgeon's experience sometimes is not enough to prevent damages as operating conditions change constantly.

2.5 Optical Coherence Tomography

As mentioned in section 2.3, current surgical microscopes also produce non-invasive imaging of the retina through Optical Coherence Tomography (OCT) [5]. OCT makes use of light waves to produce a cross-section image of the retina as shown in Figure 2. This equipment allows ophthalmologists to observe the different retina layers, measure their thickness and check for diseases. OCT is made possible because the various layers of the retina produce different responses to light waves. The OCT scanning lasts for about 5 - 10 minutes and produces a cube of images that can be sliced to obtain the cross-sections of the retina aforementioned. OCT is widely used for diagnosis of various conditions among which macular holes, macular edema, age-related macular degeneration, glaucoma and diabetic retinopathy [5]. However, since the

OCT scanner employs light waves, it is useless for conditions that prevent light to pass through the eye as, for instance, cataracts.

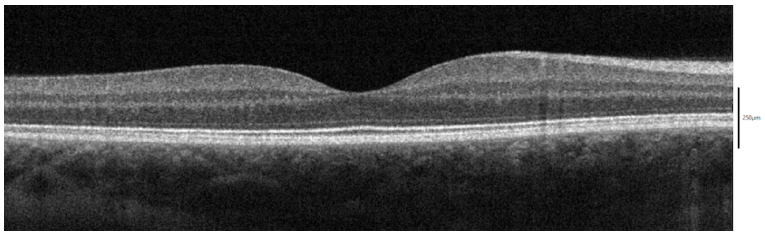

Figure 2: Optical coherence tomography scan.*

*Image by courtesy of Dr. Stephen Boppart, Biophotonics Imaging Laboratory, University of Illinois at Urbana-Champaign. EB 005221 - NIBIB, http://www.nibib.nih.gov/publicPage.cfm?section=gallery&action=desc&page=1&photo=27, Public Domain, https://commons.wikimedia.org/w/index.php?curid=11909114

2.6 Intraoperative Optical Coherence Tomography

The OCT scan described in section 2.5 is an offline methodology *i.e.*, analyzed after the surgery ends for more in-depth medical controls. This scan is surely useful for the diagnosis of eye conditions as said in section 2.5. However, an on-line OCT able to show the real-time status of the retina during the surgery is useful. iOCT, Intraoperative Optical Coherence Tomography [5], works the same as OCT but on-line during the surgeries. Instead of showing a single cross-section of the retina, iOCT shows the two cross-sections, which intersect in the point iOCT points to, allowing us to have a view along both the x and y axis through the retina.

2.7 Computer Vision

Due to a large amount of visual data available thanks to the use of advanced surgical microscopes *i.e.*, 3D videos, 3D images, OCT scans, iOCT videos and more, there have been many of computer vision applications in ophthalmology [6]. Computer Vision (CV) is the branch of Computer Science dealing with how computers can gain understanding from visual data [7] [8]. Computer vision's techniques include object detection, tracking, features detection, pattern matching, stereo vision and many more [7]. Given the type of data provided by the microscope, stereo vision is particularly attractive. There exist, in fact, various applications and research projects concerning stereo vision in ophthalmology. Stereo vision is the sub-field of computer vision that allows a perception of depth from 2D images.

2.8 Deep Learning

With the growing popularity of Deep Learning, the desire to perform the tasks mentioned in section 2.7 with Deep Learning methodologies increased. Deep Learning is the name of a family of Machine Learning algorithms employing mostly Deep Neural Networks (DNNs) [9], a modified version of standard Artificial Neural Networks (ANNs) [9] [10]. Artificial Neural Networks are statistical models used to perform a wide range of tasks, from sales forecasting to e-mail classification. Their main peculiarity is their ability to generalize on large amounts of data to perform tasks without explicit instructions. Deep Neural Networks differ from standard Artificial Neural Networks by the ability to automatically learn features from raw data instead of requiring handcrafted features by a human expert. In particular, Convolutional Neural Networks (CNNs) [11] [9], a sub-set of DNNs, leverage the mathematical operation of convolution

to extract features. They have proved to be very effective in computer vision task becoming the state-of-the-art for computer vision tasks in terms of accuracy and flexibility. For this reason, Convolutional Neural Networks found a large number of applications in ophthalmology with good results.

CHAPTER 3

RELATED WORK

The interest for Stereo Vision and depth estimation in healthcare, and in particular in ophthalmology, is not new. Having a perception of depth during critical stages of surgeries is important. For this reason, Stereo Vision has been applied in many different manners and with many different purposes. In [12], for instance, classic Stereo Vision techniques are applied to estimate the depth structure of the optic disc and from that extract useful information for the diagnosis of glaucoma. A more advanced application of Stereo Vision is a 3-D reconstruction of a scene, which produces a more concrete depth model of the scene. This is the case of [13] in which a 3-D model of the retina is reconstructed starting from stereo images.

More recently the increasing interest in Deep Learning merged with ophthalmology's needs. In [14] Convolutional Neural Network are exploited on fundus images to perform regular pixel classification and detect retinal lesions. In [15] instead, the power of U-net-like Convolutional Neural Networks is employed to estimate the depth map of the optic disc as in [12] but this time starting from monocular images. Another interesting use of Deep Learning on fundus images can be read in [16] where haemorrhages are detected from retina images with ConvNets. The compelling aspect of this work is the deep focus on data instead of the topology of the network. As said in the paper, the training performances of a network can be improved and speed-up by focusing on the learning effort of the network on the most challenging samples that usually contain a large amount of information. [17] proposes an improved system to learn

features to assess the gravity of cataracts based on deep learning. More compliant with the goal of this thesis is the work presented in [18] where a Convolutional Neural Network is employed to perform regression and estimate the coordinates of two important retinal landmarks: fovea and the optic disc. We also cite [19], which explores the effectiveness of transfer learning with and without fine-tuning on medical images that we think has some degree of interest for future work.

In conclusion we want to cite a work in which a distance and force sensing needle making use of optic fiber and OCT scans is described. In [20] an hardware device is designed, developed, and validated with a similar purposes of this thesis: avoid dangerous collision with the retina. However, while they make use of additional hardware devices, constituted by the needle described above and a special interrogator able to read the data produced through the optic fiber, we rely on stereo videos and software solutions only. These differences make our solution cheaper and easier to deploy.

CHAPTER 4

FORMAL DEFINITION AND PROBLEM STATEMENT

In this Chapter, we first define formally the statement of the problem we are facing.

We then provide all the theoretical details and introduce the technical terminology of the methodologies employed to implement our solution. In particular, we describe the computer vision techniques of stereo vision, thresholding and histogram equalization that we extensively used. We then draw a complete description of Artificial Neural Networks and Convolutional Neural Networks.

4.1 Problem Statement

The problem this work proposes to solve is the lack of reliable information about the distance of the instrument's tip from the retina surface. We are now ready to formally state the computational problem definition.

Given 1920×1080 stereoscopic frames from videos of vitrectomy surgeries with resolution 3840×1080 at 60 FPS showing the surgical instrument and the retina fundus, track the instrument's tip in the 3D space to detect and avoid collision between the surgical instrument and the retina.

In order to achieve this result, we present an automated pipeline, which locates the instrument's tip in the frames and estimates its distance from the retina. In details, our proposed pipeline:

- locates the instrument's tip coordinates (x, y) in the 2D space;
- estimates the depth of the tip *i.e.*, the z coordinate, to locate it in the 3D space;
- performs the computation in real-time to provide ophthalmologists with the position of the instrument's tip in the 3D space.

Fundus depth estimation, retina collision avoidance and retina landmarks localization have already been attempted applying both standard Computer Vision and Deep Learning. However, the novelty of our work resides in the purpose and the unique combination of these techniques.

4.2 Machine Learning

Machine Learning (ML) is a sub-field of Computer Science that employs algorithms and statistical models to build software that can perform tasks of different kinds without predefined, explicit instructions [9] [11]. The core of learning in Machine Learning is data. Thanks to statistical models, Machine Learning can learn to perform tasks by generalizing on the training data.

Machine Learning paradigms differ from each other depending on the type of task and data available. The three main paradigms of Machine Learning are Supervised Learning, Unsupervised Learning, and Reinforcement Learning. **Supervised Learning** is the most used and most mature of the three. It makes use of *annotated data i.e.*, data with multiple input variables and one or more desired outputs. Input variables are also called features, attributes or independent variables; output variables are also called targets, labels or dependent variables. Supervised learning tries to approximate the unknown function that relates input and output variables. Supervised Learning tasks include regression and classification.

Given an annotated dataset, the task of estimate a real, continuous value is called *regression*. This is the case of estimating coordinates in an image or predicting a selling price. The most simple form of regression is Simple Linear Regression [11]. It concerns a single input variable and a single output variable and uses a linear function of the input variables to approximate the relation between input and output. Due to the number of variables, also called the dimension of the dataset, the Simple Linear Regression is easy to be visualized with x and y in the Cartesian coordinate system. More advanced regression approaches include Multiple Linear Regression [11], which makes use of multiple independent variables, and Polynomial Regression [11], which employs functions with degree > 1 to approximate the relation between input and output.

To learn the unknown function, regression employs a loss function and a set of weights. The loss function summarizes the error between the output value produced by the regression model and the real, expected output value. To improve the performance of the model, the loss function has to be minimized *i.e.*, the global minimum of the function has to be found, corresponding to the minimum error between the inferred and the real output. The weights are used to adjust the statistical model to fit the data better and decrease the value of the loss function. Classic loss functions for regression tasks are Mean Squared Error (MSE) [9]

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.1}$$

Root Mean Squared Error (RMSE) [21]

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.2}$$

Mean Absolute Error (MAE) [9]

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.3}$$

Residual Sum of Squares (RSS) [22]

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{4.4}$$

Coefficient of Determination R2 [23]

$$1 - \frac{RSS_{tot}}{RSS}, \tag{4.5}$$

where

$$RSS_{tot} = \sum_{i=1}^{n}(y_i - \overline{y_i})^2 \tag{4.6}$$

and

$$\overline{y_i} = \frac{1}{n}\sum_{i=1}^{n} y_i, \tag{4.7}$$

where y_i is the produced output, $\hat{y}_i$ is the expected value, and n is the number of samples.

To minimize the loss function multiple approaches are possible. They can be divided into:

- direct approaches, among which Least Squares and Gradient Descent;
- discriminative approaches, including Maximum Likelihood;
- generative approaches.

For the purpose of this thesis, here we analyze direct approaches only. Given the loss function L, function of the weights w, Least Squares looks for the point where the gradient of L is null ($= 0$) and their eigenvectors are positive (the function is convex). To perform this computation, Least Squares [11] compute the first and second derivative of L and look for the weights vector w that makes the first derivative null and the second derivative positive. For large datasets, minimization with Least Squares is not feasible as the number of parameters explodes. In this case, a iterative, on-line algorithm called Gradient Descent [11] is used. After defining the loss function L as a sum over samples, Gradient Descent computes the gradient of the loss function one sample at a time and updates the weights vector w by following down the slope of the gradient with a fixed step η, called learning rate. See parameters updating rule of Gradient Descent in equation Equation 4.8.

$$\mathbf{w}^{k+1} = \mathbf{w}^{k} - \eta^{k} \cdot \nabla L \tag{4.8}$$

where k is the iteration. There exist several variations of Gradient Descent, which differ each other in the updating rule. Some of them vary the magnitude of the learning rate, others add some term in the equation, others make use of decaying running averages of the gradient.

When the dependent variable is a discrete value, the task is called *classification* because the discrete values of the dependent variable can be seen as the labels of classes to which the samples in the dataset can belong to. This is the case of spam/not-spam email classification or ML-based diagnosis. Classification can be binary when there exists only two classes in the

dataset, or multi-class when more than two. Unlike regression that is said to be linear if the function relating input and weights to the output is linear, Classification is said linear if the decision boundary is linear *i.e.*, if the surface dividing the samples is a line. If the input space is not linearly separable, as the XOR-problem, a non-linear separation boundary for classification is needed. As for regression, the learning performance of a classification model is measured with a loss function. A typical loss function for classification tasks is the Cross-Entropy function [11]

$$-\sum_{i=1}^{n}(\hat{y_i}\ln y_i + (1-\hat{y_i})(1-\ln y_i)) \tag{4.9}$$

where y_i is the produced output, $\hat{y_i}$ is the expected value, and n is the number of data points. An example of a linear model for classification is the perceptron [11]. The most widely used implementation of binary classification is Logistic Regression [11]. Like regression, it outputs a real value that, however, is interpreted as the probability of a sample to belong to one of the two classes, so that a sample is labelled with the class generating the highest probability. Logistic Regression employs Sigmoid as activation function. More advanced classification algorithms exist, such as Decision Trees [9] or Naïve Bayes [9], the latter employing *a-posteriori* probability derived by the samples in the dataset to classify never seen samples. What makes Naïve Bayes *Naïve i.e.*, simple, is the assumption that the samples are conditionally independent and identically distributed, which is a quite strong assumption.

Non-linear regression and classification models are more powerful than linear ones as they can fits data better. For this reason, Artificial Neural Networks, which are non-linear models

described in details in the subsequent sections, are widely used. They can, in fact, effectively approximate non-linear functions.

Unsupervised Learning employs data without a target value or a label and it is used to learn previously unknown patterns of the data. The typical unsupervised task is represented by clustering. In clustering, the algorithm tries to divide the examples into clusters based on intrinsic similarities of the data itself and not based on a provided label as in classification. A well-known clustering algorithm is K-means [11]. In K-means a fixed number of cluster K is chosen a-priori and then the algorithm tries to minimize the intra-cluster variance of samples while keeping a high the inter-cluster variance.

Finally, in **Reinforcement Learning** an agent is trained to perform actions into an environments to maximize a reward. This process is usually modeled as a Markov Decision Process (MDP) [10].

4.2.1 Artificial Neural Networks

Artificial Neural Networks [9] [10] are a Machine Learning model inspired to the learning process of the human brain. They are built with layers of perceptrons, a.k.a. neurons, which constitute the basic structure of learning in ANN. Typically, the topology of an ANN is the following, as showed in Figure 3:

- an input layer with as many neurons as the number of input variables;
- one or more hidden layers with multiple neurons;
- an output layer with one or more neurons, depending on the number of output needed.

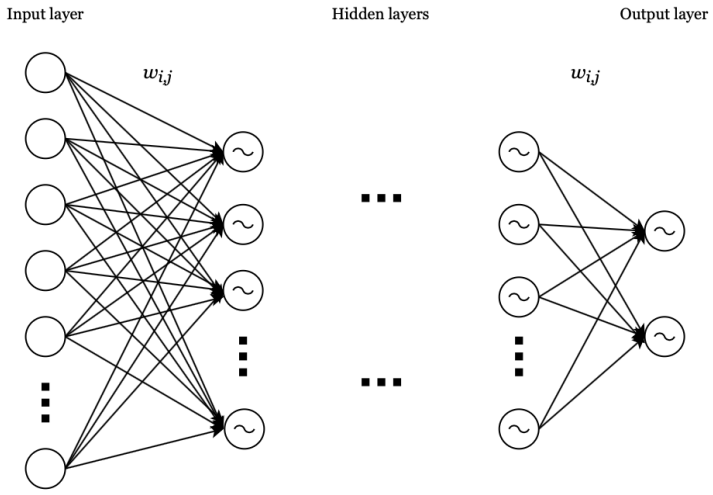

Figure 3: Topology of a standard feed forward artificial neural network.

These layers can be connected in a forward way to build Feed-Forward Neural Networks (FFNN) [11], or with cycles to build Recurrent Neural Networks (RNN) [9]. The latter is usually employed on data with some kind of time dependency.

The output of each neuron in each layer is feed as input to all the neurons in the subsequent layer. These interconnections between the layers are characterized by weights $w_{i,j}$, which constitute the *trainable* part of an ANN. Updating the weights makes possible to change the final output of the ANN and so to minimize the difference between the output of the model and the target value. This difference is called error and, as already said in section 4.2, is measured with a loss function, which is a function of the weights and the input. This said, *training an ANN* means to find the weights' vector such that the loss function value reaches one of its minima.

To do so, we must be able to compute the minima of the loss function. However, loss functions employed in ANNs are usually complex, n-dimensional functions with many parameters and very difficult to study. For this reason, ANNs employ the Gradient Descent approach, described in section 4.2. With Gradient Descent, starting from a random initial point corresponding to a random initialization of the weights in the ANN, we follow down the slope of the loss function's gradient to get closer to a minimum, as showed in Figure 4. However, this algorithm doesn't assure to reach the global minimum of the loss function as reaching a minimun strongly depends on the starting point and the nature of the loss function. For this reason, there's always the danger of ending in a local minimum or, even worst, to never get to a minimum. Due to the complexity of the layered structure of a ANN the numerous weights are updated with back-propagation.

As mentioned closing section 4.2, Artificial Neural Nets are considered a powerful tool in Machine Learning because of their ability to effectively approximate very complex, non-linear functions. This effectiveness is made possible thanks to the introduction of non-linear activation function in the neurons. See figure Figure 5. Activation functions [9] [11] are functions employed inside each neuron to filter the linear combination of input and weights that each neuron receives from the previous layer. Since the output of an ANN is defined by linear combinations between weights and input and subsequent composition of the activation function in each neuron, the introduction of at least one non-linear activation function ensures to approximate a non-linear function. This happens because the composition of linear functions only is still a linear function.

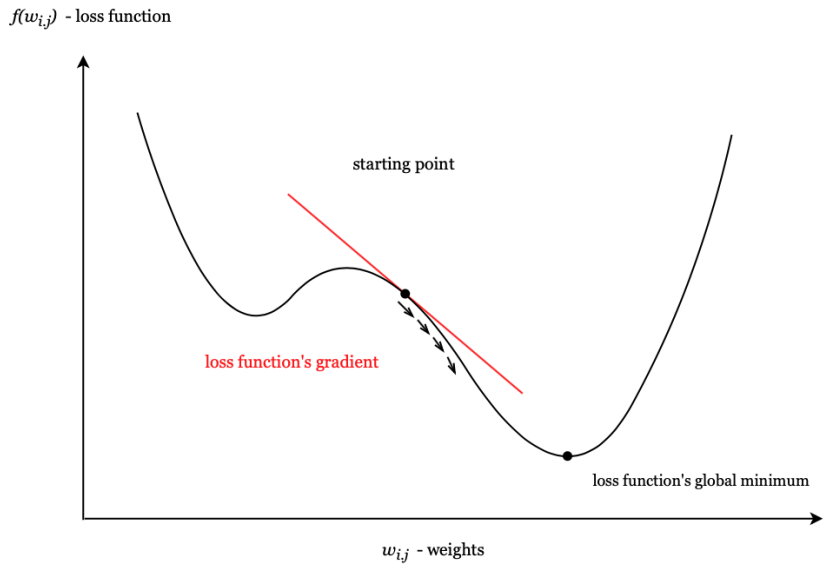

Figure 4: Gradient descent idea.

There exist several activation functions employed in Artificial Neural Networks, which vary based on the task. We mention:

- the linear function, which is still used for regression tasks only in the output layer to output real values;
- the ReLU (Rectified Linear Unit) used both in the output layer to produce non-negative real values and in hidden layers as non-linear function;
- the sigmoid function, particularly useful for classification tasks as its output ranges between 0 and 1.

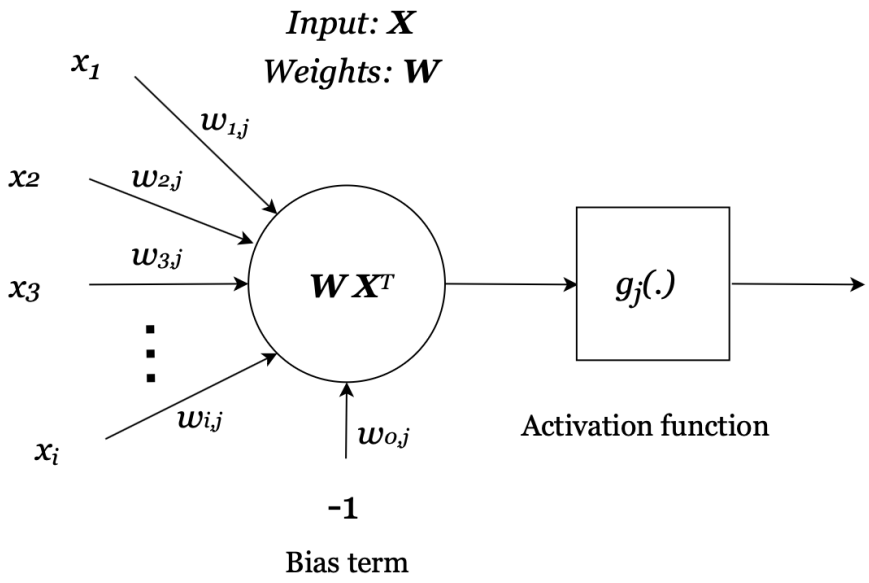

Figure 5: A model of an artificial neuron.

However, a well-known issue of high dimensional functions is a phenomenon called overfitting. Overfitting happens when a Machine Learning model, "learns by heart" the training set by perfectly fitting it so that its generalization capabilities on never seen samples are reduced. Overfitting is generally due to lack of data and large value of the weights. Luckily some techniques have been developed to prevent overfitting. First of all, there exists dropout, which simply randomly disable neurons' interconnections in ANNs with probability p to enhance generalization. Moreover, there exists regularization, which is embedded in loss function to penalize large weights during updates and once again improve the generalization capabilities of the network. Regularization can be added to the loss function as *Ridge regularization* (or

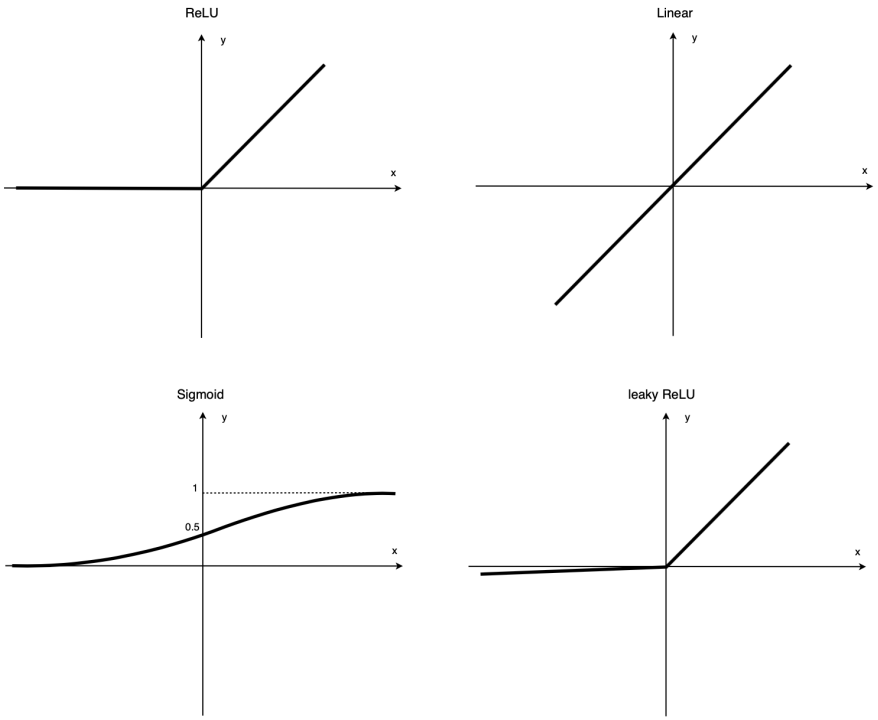

Figure 6: Some activation functions.

weight decay) [11], which adds a penalty equal to the squared norm of the weights, or as *Lasso regularization* [11], which adds a penalty equal to the norm of the weights. Lastly, the more trivial way to prevent overfitting is be in possession of a large amount of data. This method may seem trivial but with Supervised Learning algorithms described in section 4.2 this would require to annotate a huge amount of data. Together with overfitting, a less problematic phenomenon called underfitting can appear. Underfitting occurs when the network, or the Machine Learning model, is not able to sufficiently fit the data.

4.3 Deep Learning

Deep Learning (DL) [9] is the name of a set of Machine Learning (ML) methods employing mostly Artificial Neural Networks (ANN) to solve regression, classification and more advanced tasks such as segmentation and tracking in fields among which Computer Vision (CV) [9], finance, and Natural Language Processing (NLP) [9]. Deep Neural Networks (DNN) [9], unlike standard Artificial Neural Networks described in section 4.2.1, also referred to as *shallow* ANN, are characterized by a high number of layers, the depth of an ANN. Moreover, another peculiarity of DNNs is the employment of an automatic features extraction section. In fact, unlike standard Artificial Neural Networks, which are feed with features hand-crafted by data engineers, DNNs receive raw data and can learn a deep understanding of features thanks to the embedded features extraction section.

4.3.1 Convolutional Neural Networks

Convolutional Neural Networks [11] [9], in short ConvNets or CNNs, are a particular type of Deep Neural Network that make use of the mathematical operation of convolution [9] to

extract features from the data. Convolution makes use of a convolution kernel *i.e.*, a matrix, containing values called weights and shifting on the input data. The kernel can have different size and even different dimension. For instance, for NLP tasks in which the input data are 1-D sequences of words, the convolution kernel is a 1-D matrix *i.e.*, an array. Otherwise, for Computer Vision tasks where the input data are images the kernel may be a 2-D matrix with height and width or even a 3-D matrix with height, width and depth to take into account the RGB channels of the image. The weights inside the kernel can be trained to learn features from the input, similarly to interconnections' weights for ANN. Following, we consider the case of ConvNets dealing with images as this is our case of interest.

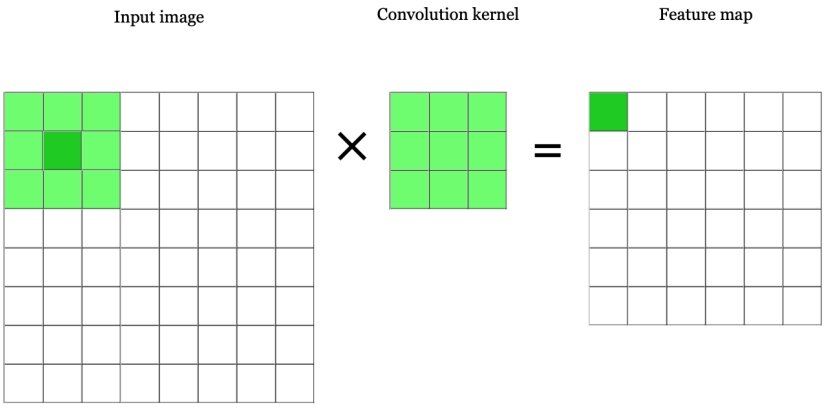

Figure 7: Convolution.

In this scenario, the kernel is used for convolution by computing the element wise product between a kernel's weights and the underlining pixels for each kernel position on the input image and then summing up the result (dot product). The result of the *dot-product* is feed into an artificial neuron with an activation function. Each filter has a neuron associated to it. The matrix of all the outputs from a neuron for all the positions of a kernel on the input image is called feature map or activation map. A feature map represents the features learnt by the network for a particular input with a particular kernel. As can be seen from Figure 7, due to the size of the kernel, convolution consumes some pixels on the border of the input image. These pixels get lost unless a padding is used on the input image. Usually, padding consists of adding zeroed pixels around the image. The size of the padding is relative to the size of the kernel. In Figure 8 the padding is 1 since the kernel size is 3×3. Another little variation to standard convolution is the stride. Stride is the step of the convolution kernel on the input image, as can be seen on Figure 9. It is used to avoid overlap the kernel of the image and to reduce the amount of data produced by convolution.

Thanks to convolution, ConvNets can learn spatial features from the images *i.e.*, features belonging to the 2-D representation of the images. Similarly to ANNs, convolution represents in every way a hidden layer of the network.

In this sense, ConvNets usually include multiple convolutional layers each of them having multiple convolutional kernels as shown in Figure 10. Each kernel corresponds to a single artificial neuron and a single feature. These kernels are called filters. A typical example of features are edges: a ConvNet can learn to locate edges in images by training a filter to

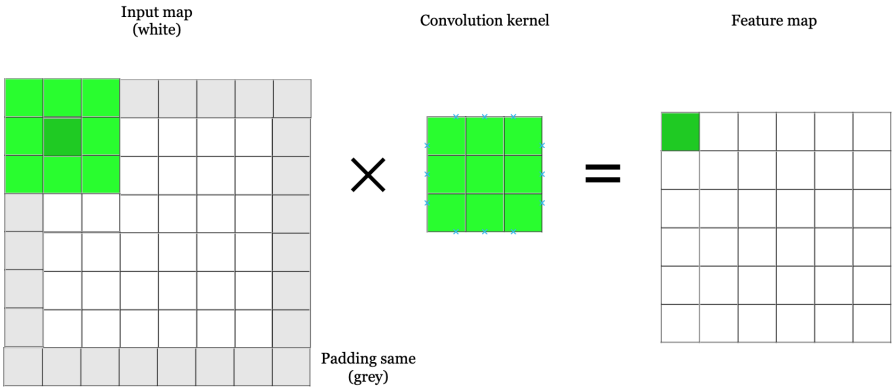

Figure 8: Convolution with padding.

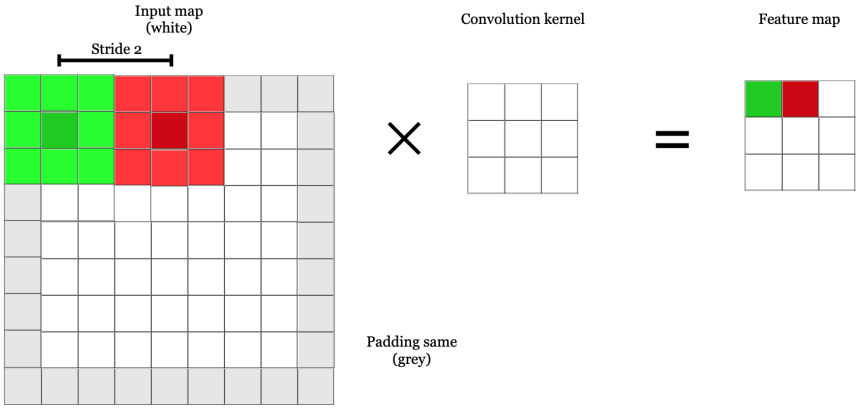

Figure 9: Convolution with stride.

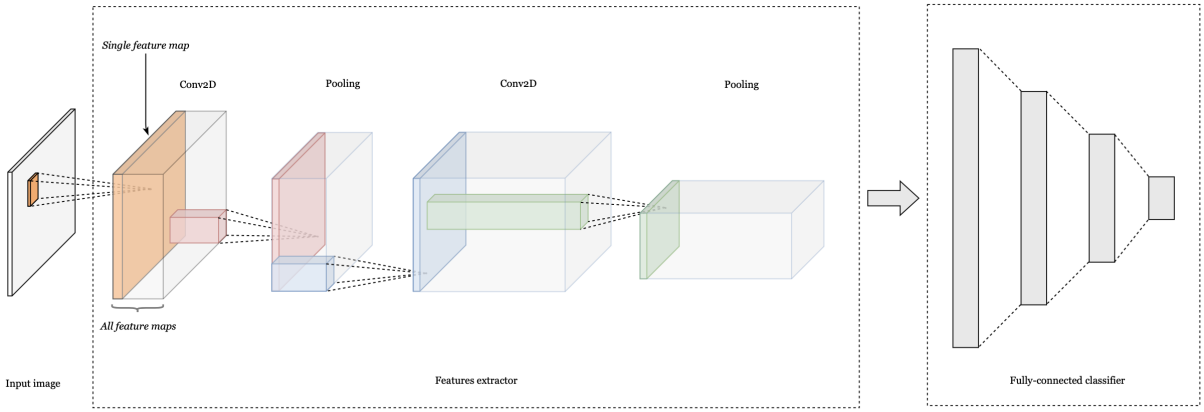

Figure 10: Structure of a convolutional neural network.

detect areas of transition in pixels' intensity. Thanks to their multiple sequential layers of convolution, CNNs learn a hierarchy of features. This hierarchy means that filters in the first convolutional layers learn low-level features while filters in the last convolutional layers learn high-level features. Sometimes these features are easily understandable by humans, as for edges and shadows, sometimes they are not understandable and they are said to lack in interpretability. Since the amount of data produced with the different filters by convolution can be considerable, ConvNets usually employ an additional layer to summarize and compress data called Pooling [9]. Pooling gathers together multiple values of all the feature maps of one convolutional layer into a single value to be streamlined to the next one. Normally polling is performed with max or average.

Convolutional Neural Networks showed to be very reliable and powerful for automatic features extraction and learning relying on some interesting properties. First of all, considering

each single pixel as an input feature would lead ANNs to suffer from the Curse of Dimensionality [11] *i.e.*, the case in which the considered dataset contains way more attributes (dimensionality of the dataset) than data points, leading to difficulties in learning. In fact, considering a low resolution RGB image 512×512 leads to $512 \times 512 \times 3 = 786,432$ input variables. Moreover, the use of a convolutional kernel assures to take into account the spatial structure of the image (locality), relating close pixels each other. Nevertheless, pooling grants some degree of shift invariance.

4.4 Computer Vision

Computer Vision, in short CV, is the branch of the Computer Science dealing with how computers can gain understanding from visual data *i.e.*, images and videos. It gathers techniques and algorithms for images and videos handling, Image Processing, Video Tracking, Object Detection, Artificial Intelligence, Features Detection & Matching, Stereo Vision, and many more [24].

These techniques are usually exploited to automate visual tasks that can be performed by humans.

4.4.1 Stereo Vision

Stereo Vision [24] is the sub-field of Computer Vision that allows transforming a scene from the 2D world to the 3D world. Unlike standard image and video capturing that makes use of a single camera, stereo vision needs two or more cameras to be able to compute the depth of the captured scene. This availabity of additional data enables further computation and processing that single-camera visual data don't allow. Following, we analyze the special case

of two stereo cameras since it is the most common case in practice and it is the case of our surgical microscope.

We call left-view the image captured by the camera on the left and, similarly, right-view the image captured by the camera on the right. To be able to derive depth of single points and, consequently, of entire objects, first of all, we must be able to match corresponding points between the left and the right view. This is the so-called Correspondence Problem [24] *i.e.*, given a point x in the left view find the corresponding point x' in the right view. To tackle this problem the Epipolar Geometry [24] comes to the aid.

4.4.1.1 Epipolar Geometry

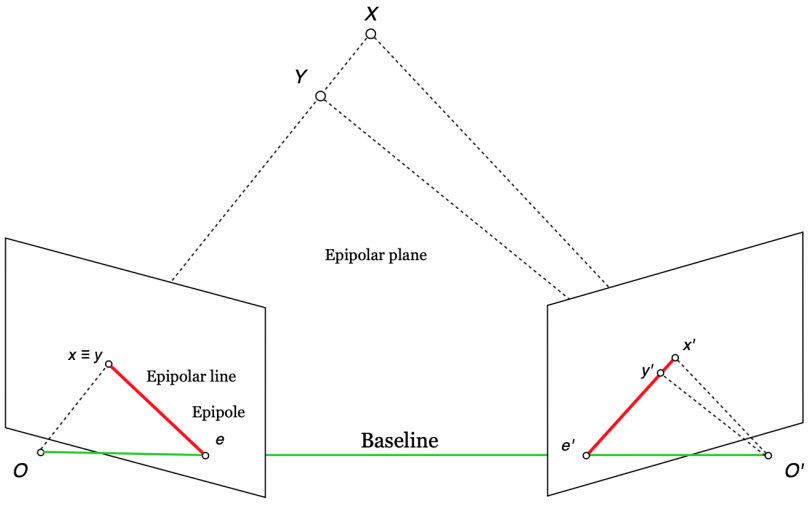

Figure 11: Epipolar geometry.

The Epipolar Geometry is the geometry of a stereo vision scene that gathers all the mathematical relations between the 3D points and their corresponding projections in the 2D space. Let's consider a point X in the 3D space and its 2D-projections on the left and the right view, respectively, x and x' with camera centres O and O'. We call *baseline* the line connecting the two camera centres; the *epipolar plane* is the 1-D plane containing the baseline and the point X; the *epipoles* are the intersections of the baseline with the image planes of the two cameras; the *epipolar lines* for the point X are the intersections of the epipolar plane with the image planes. Epipoles can happen to lie outside of the image planes if the rotation between the left and the right camera doesn't allow an intersection. This said, potential matches for point x on the left view must lie on the epipolar line corresponding to point X in the right view and vice-versa. This is called Epipolar Constraint and it deeply facilitates the correspondence problem as we have to check a very limited number of points for matches instead of looking at the entire image.

4.4.1.2 Image Rectification

The easy, ideal stereo vision setup consists of cameras with image planes parallel one to the other, parallel to the baseline and with camera centres at the same height. In this simplified case the epipolar lines are always horizontal, parallel with the horizontal edges of the images. In the more general case images planes are not parallel and the above-mentioned property of the epipolar lines doesn't hold anymore. In this case, we must apply a procedure known as Rectification [7] to remove the distortion due to the perspective. Rectification consists of re-projecting the two image planes to a common plane parallel to the baseline. After this

transformation, the geometric relations are the same as the aforementioned ideal case with parallel image planes.

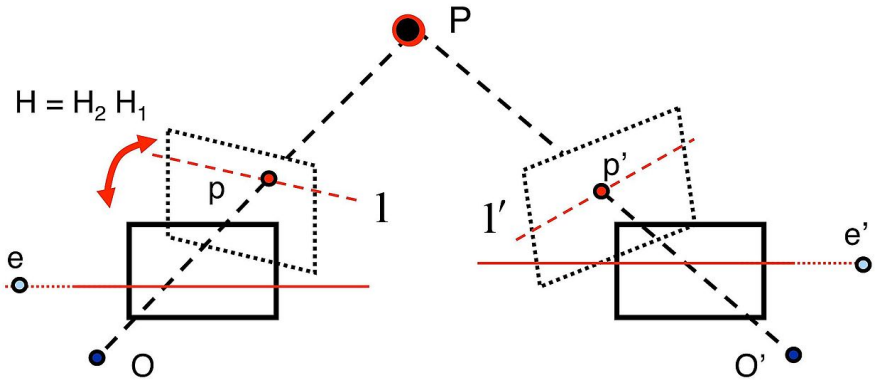

Figure 12: Rectification example.*

*By Silvio Savarese - Lecture presentation for computer vision, Public Domain, https://commons.wikimedia.org/w/index.php?curid=37248047

4.4.1.3 Camera Calibration

Real-world lenses are not perfectly flat but have some degree of warp. This characteristic can affects the final images with different degrees of distortion, higher near the edges of the images, that can make straight lines look bent. Moreover, in 2D images, all distance must be measured in pixels instead of length units as all real-world references have been lost.

Camera Calibration [24] is the process that allows us to compute the so-called intrinsic and extrinsic parameters of a camera and overcome the above-mentioned issues. Intrinsic parameters

are parameters depending only on the camera's manufacturing. They include focal length f, camera centres and image format. Instead, extrinsic ones are those parameters depending on the relative position of one camera w.r.t. the other and they include a translation factor T and a rotation factor R.

A standard procedure to estimate these parameters makes use of a known pattern with known dimension and numerous key points, such as a chessboard. This additional knowledge allows to transform the known pattern from the 3D to the 2D space of an image and derive the parameters that allowed that transformation.

In our case, we were not able to calibrate the stereo cameras as we received the stereo videos only without information about the stereo cameras that recorded the surgery and their intrinsic parameters.

4.4.1.4 Disparity Maps

The disparity map is an gray scale image, which contains the disparity value for all the pixels in the original stereo images. The disparity [24] [7] of a pixel is defined as the difference in the x coordinate of matching points between the left and the right image. It is important to notice that we can compare the x coordinate only thanks to the epipolar constraint [24] that limits our search on the epipolar lines and thanks to rectification that makes epipolar lines horizontal. This way matching points have the same y coordinate. Thank to disparities it is then possible to compute depth through the baseline and the focal length with

$$z = \frac{B \cdot f}{disparity} \tag{4.10}$$

where f is the focal length of the camera pair, B is the baseline of the setup, and z in the depth. We now show as this formula can be derived thanks to the Epipolar Geometry. Let's

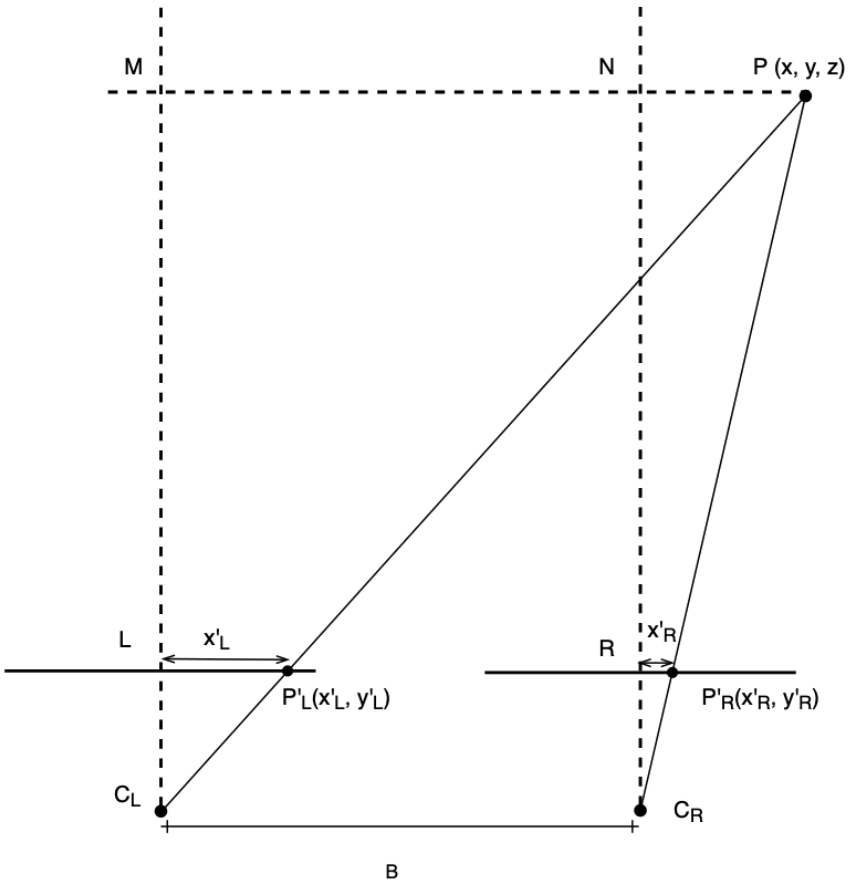

Figure 13: Rectified stereo vision setup.

consider the setup in Figure 13 with a 3D point P with coordinates (x, y, z), cameras centers

C_L and C_R, images centers L and R, cameras axis M and N, and projections p'_L and p'_R of the P. We fix the origin of the coordinate system in the left camera center C_L. We can consider the similar triangles PMC_L and $p'_L LC_L$ and derive

$$\frac{x}{x'_L} = \frac{z}{f} \tag{4.11}$$

where $PM = x$, $MC_L = z$, $p'_L L = x'_L$ and $LC_L = f$. Similarly with similar triangles PNC_R and $p'_R RC_R$ we derive

$$\frac{x - B}{x'_R} = \frac{z}{f} \tag{4.12}$$

where $PN = x - B$, $NC_R = z$, $p'_R R = x'_R$ and $RC_R = f$. From Equation 4.11 we obtain

$$x = \frac{z \cdot x'_L}{f} \tag{4.13}$$

that substituted in Equation 4.12

$$\frac{z \cdot x'_L}{f} - B = \frac{z \cdot x'_R}{f} \tag{4.14}$$

Cleaning up equation Equation 4.14

$$\frac{z \cdot x'_L}{f} - \frac{z \cdot x'_R}{f} = B \tag{4.15}$$

$$\frac{z}{f}(x'_L - x'_R) = B \tag{4.16}$$

We finally obtain that

$$z = \frac{B \cdot f}{(x'_L - x'_R)} = \frac{B \cdot f}{d} \tag{4.17}$$

where $x'_L - x'_R$ is the disparity d of point P, z is the depth of point P, B is the baseline of the stereo setup, and f the focal length. From Equation 4.17 it can be noticed that disparity must be large for object close to the camera and small for objects far away from the camera. Disparity is in fact inversely proportional to depth.

4.4.2 Thresholding

Image thresholding [24] is a Computer Vision technique applied to greyscale images to obtain a binary image that usually divides the foreground from the background. This technique is widely used in many computer vision sub-fields such as segmentation. Simple image thresholding fixes a unique intensity threshold so that pixels with intensity below this threshold are set to 0 (black) and pixels with intensity above the threshold are set to 255 (white). A big limitation of this simple thresholding algorithm is the need of a human expert setting the global threshold for all the pixels. Nevertheless, a value for the threshold can be automatically chosen by computing the average intensity value of the greyscale image and iteratively tune this value. However, simple thresholding works better with clear foreground-background separation and good light conditions.

A more advanced algorithm is Adaptive Thresholding [25]. Instead of using a global thresholding value for all the pixels, Adaptive Thresholding computes a threshold for each pixel in the image based on the intensity values of its neighbours. It employs a mask around each pixel to compute a weighted mean. Both the size of the mask and the averaging method can be tuned.

An even more advanced algorithm to perform automatic thresholding is Otsu Thresholding [26]. It is a histogram-based method meaning that it leverages the image histogram to choose a proper value for the threshold. Otsu Thresholding tries to minimize the intra-class intensity variance of pixels by looking at the probability of each intensity value in the histogram. The algorithm is the following:

- compute the image histogram and the probability $P(i)$ of each intensity value i;
- for each threshold value t between 1 and 255, compute

$$\sigma^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{4.18}$$

  where $\sigma^2(t)$ is the intra-class variance of two classes expressed as weighted sum of the variance of the two classes, $\sigma_1^2(t)$ is the variance of the first class, $\sigma_2^2(t)$ is the variance of the second class and

$$q_1(t) = \sum_{i=0}^{t} P(i) \tag{4.19}$$

$$q_2(t) = \sum_{i=t+1}^{255} P(i) \tag{4.20}$$

- the Otsu's threshold corresponds to the value $\bar{t}$ producing the highest intra-class variance $\sigma^2(\bar{t})$.

Otsu Thresholding works particularly well with bi-modal images *i.e.*, images whose histograms present two clearly-separated peaks. However, when the valley between the two peaks in the

histogram is corrupted, due to noise or small foreground area w.r.t. the background, Otsu Thresholding presents some limitations in computing the right threshold value.

4.4.3 Histogram Equalization

Histogram equalization [7] is a computer vision technique used to adjust the contrast of an image. It is used to increases the contrast between objects in the image, especially when the foreground and the background are both dark or bright. It employs the image histogram to re-distribute large intensity peaks across the entire image. Histogram equalization is widely used for medical images that are usually under or over-exposed, such as x-ray scans. A more advanced histogram equalization technique exists, called Adaptive Histogram Equalization (AHE) [7]. Differently from standard histogram equalization, Adaptive Histogram Equalization computes several sub-histograms of the image, each one corresponding to a different area of the input image and it then applies local histogram equalization. This approach helps enhance local contrast and properly enhance edges in different regions of an image. A known issue of AHE is its tendency to amplify little amounts of noise in local, homogenous patches. This happens because the outcome of histogram equalization re-distributes peaks over the considered pixels. To avoid this issue, a slight variation of AHE has been developed called Contrast Limited Adaptive Histogram Equalization (CLAHE) [7]. CLAHE limits the amplification of noise in homogenous patches by clipping high peaks of the histogram.

CHAPTER 5

METHODS

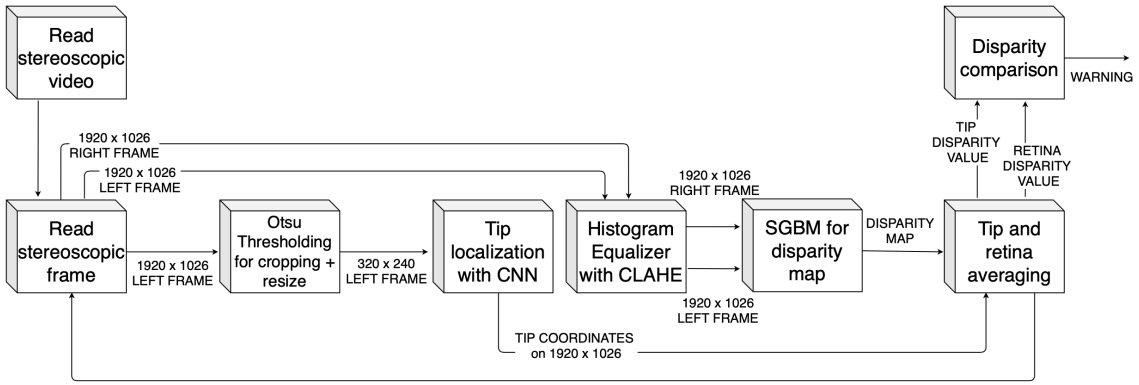

Figure 14: Pipeline overview.

5.1 Our pipeline

Our proposed solution consists of an automated pipeline employing several Computer Vision [24] [7] techniques and a Convolutional Neural Network [11] [9] to provide ophthalmologists with additional information about the relative distance of the tip of the surgical instrument to the retina.

In detail, the pipeline reads a single stereoscopic frame from a stereoscopic video of ophthalmic surgery. The current pipeline works on recorded videos for testing reasons but it can

be easily modified to work on a stream of data. It then pre-processes the stereoscopic frame to split left and right view and remove from each frame the logo of the framework employed by the surgical microscope to encode the stereoscopic video. At this point, a copy of the left frame is cropped to 1368×1026 to centre the region of interest (ROI) represented by the circular view of the fundus and remove the areas of black pixels near the edges that are useless. This step is performed employing Otsu Thresholding. The cropped left frame is also resized to 320×240. Once the left frame has been resized, it can be feed to our Convolutional Neural Network to locate the instrument's tip in the scene, if any. The outcome of this step is a tuple containing the coordinates of the instrument's tip in the left frame. Now, both the original left frame and right frames are pre-processed with Contrast Limited Adaptive Histogram Equalization just before computing the disparity map of the scene with Semi-Global Block Matching. Since disparities are independent of the camera's intrinsic parameters, we do not perform the transformation from disparity to depth as we do not want to be dependent on the microscope employed and we always work with relative values. The output of this step is a disparity map of the entire scene where landmarks are in the same coordinates system of the left frame. This is because the stereo matching algorithm uses the left frame as the reference. This implementation detail allows applying the coordinates estimated in the previous step to look for the disparity values of the pixels belonging to the instrument's tip and the background retina. To perform this computation, an average disparity value of the tool's tip is computed by averaging the disparity values around the estimated coordinates with a mask and Gaussian weights. A similar computation is performed to compute an average disparity value for the retina. The difference is the size

of the mask used, which is larger, and the weights, which should be such to exclude the pixels belonging to the instrument since it also appears in this mask. With a unique disparity value for both the instrument's tip and the retina, we perform a comparison. Warnings are issued if the comparison outputs a value below a fixed safety threshold. To provide a retina average disparity value more resilient w.r.t. noise and available even when the retina results occluded, we store the most recent disparity values estimated for the retina and compute their running mean. We discard all values distant from this mean as outliers. This approach is possible since the retina surface does not move significantly during the entire video and, in particular, among frames. On the other hand, this same approach cannot be implemented for the instrument's tip as the latter moves much more during the surgeries.

5.2 Otsu Thresholding

In our pipeline, we employed Otsu Thresholding to filter the circular view of the fundus, which is lit by a surgical torch from the areas of black pixels around it. The reason why we decided to use thresholding in detecting and cropping the region of interest in our frames is that the surgical frames have large areas of black pixels and a lit, circle area showing the retinal fundus. After analyzing the data, we figured out we could have leveraged an intensity threshold to segment the region of interest through thresholding. Moreover, exploring the strengths and flaws of different thresholding algorithms described in section 4.4.2, we realized Otsu Thresholding was the right one to employ in our scenario. Basic and simple Adaptive Thresholding showed to be too unreliable in finding a proper threshold value. On the other hand, the Otsu's Method has two main characteristics explained in section 4.4.2 that drove my

choice. First of all, Otsu Thresholding does not require a hardcoded, human-tuned threshold value as it can automatically choose the most suitable value based on the image histogram. This was important as we wanted our pipeline to work seamlessly on any surgery video. Moreover, Otsu's Method is known to work well on bi-modal images *i.e.*, images with two well-defined peaks in the image histogram. This is the case of our frames which, as described above, can be easily separated in completely black pixels and coloured ones.

5.3 Contrast Limited Adaptive Histogram Equalization

As mentioned in Chapter 1, the light conditions of ophthalmic surgeries are inconsistent: some areas appear well lit by the surgical torch while others are dark or affected by glares. Stereo matching algorithms, and so disparity maps quality, can be deeply affected by light conditions of the image. This said the available techniques were explored and we found that histogram equalization [7] can adjust contrast in images and improve stereo matching algorithms. Among the histogram equalization algorithms, CLAHE (Contrast Limited Adaptive Histogram Equalization) [7] was finally chosen because it does not suffer from noise-amplification issue of AHE (Adaptive Histogram Equalization) [7] but still applies local equalization Moreover, it is widely used in medical imaging as this class of images are usually over or under exposed.

5.4 Semi-Global Block Matching

As the goal of our work is to detect and avoid retinal collisions, we initially planned to estimate the relative distance between the surgical instrument and the retina surface in two ways: leverage the 2D distance between the instrument's tip and its shadow on the retina, suggested by our ophthalmologist expert as the most widely used visual cue currently employed in the

operating room as mentioned in section 2.4, and exploiting the advanced visual data collected during the surgery *i.e.*, the stereoscopic videos, described in section 2.3. Eventually, segmenting the instrument's shadow in the frames and computing the Euclidean distance between the tip of the shadow and the tip of the instrument resulted to be unreliable and cumbersome. The difficulties arose mainly in locating the shadow in the frames. It is, in fact, not always present or its shape not always sharp as these properties are strongly correlated with the position of the surgical torch. For all these reasons and the availability of stereoscopic videos, we decided to face the problem by employing stereo vision. While exploring and working with stereo vision, we realized that, as we miss the intrinsic parameters of the stereo cameras described in section 4.4.1.3 and we are interested in a relative distance measure instead of an absolute, real distance value, we could have worked with disparity maps that are the last step before computing depth. Disparity values in disparity maps belong to the same scale, meaning that if two pixels have the same disparity value their depth would be the same. This allowed us to compare the disparity values of pixels without computing their depth. Finally, when we had to choose a stereo matching implementation to compute disparity maps we took into consideration several stereo matching algorithms but, after we have read some documentation and have tested them on our data, we choose to employ SGBM (Semi-Global Block Matching) [27] because of its well-known tradeoff between runtime and quality of the produced disparity maps. A stereo matching algorithm is needed to solve the Correspondence Problem mentioned in section 4.4.1 and find corresponding points between the left and the right frame. The peculiarity of SGBM is that it looks for corresponding points within a subset of the image. Given a pair of rectified

images and point p in the left image with coordinates (x, y), SGBM looks for the match p' in the right image as

$$\{x' \geq x \wedge x' \leq x + D\} \tag{5.1}$$

where D is the maximum allowed disparity. This limited search space reduces dramatically the run-time.

5.5 Convolutional Neural Network for Instruments' Tip Localization

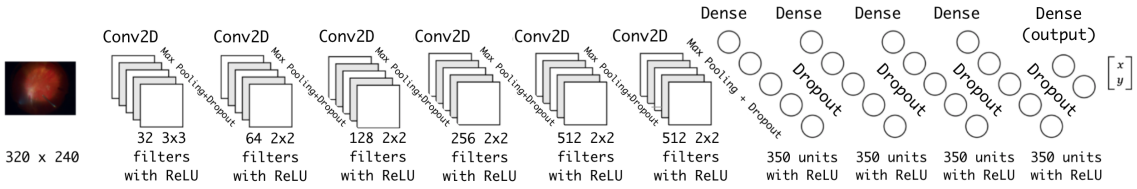

Figure 15: Structure of our CNN.

Before starting the long process of building a Convolutional Neural Network, we attempted various standard Computer Vision techniques to locate the tip of the surgical instrument. However, a common issue of all the attempted techniques was the lack of generalization and the sensitivity to data variance that CNNs, with their intrinsic properties, can overcome. The techniques we attempted include Hough Line Transform [7], a computer vision algorithm used to detect straight lines in images. We employed Hough Line Transform to detect the two straight lines outlining the edges of the surgical tool and then approximate the location of

the instrument's tip as the intersection point of these two lines. However, this technique was very unreliable due to the high encoding compression of the frames making the instrument's edges dull and blurred. Moreover, not all the surgical instruments in the videos are straight, making locating the instrument's tip with the intersection point of the Hough lines unfeasible. Nevertheless, some frames also contain a very noticeable instrument's shadow that Hough Line Transform was used to detect as a straight line in place of the instrument's body. Some examples of the performance of Hough Line Transform can be seen in Figure 16

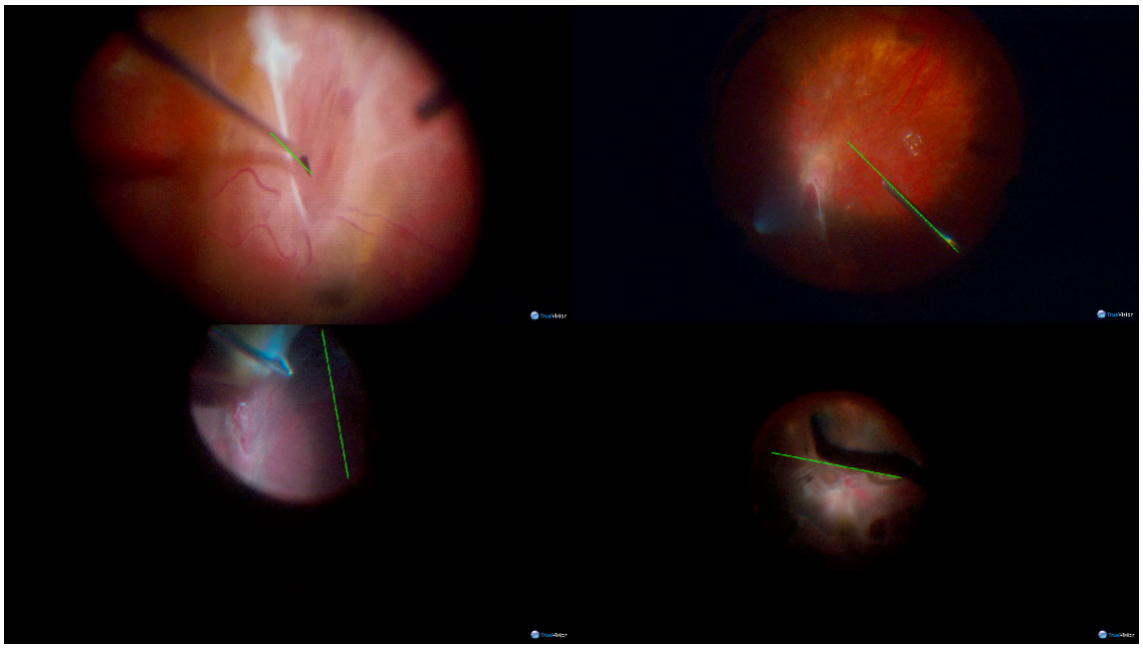

Figure 16: Hough Line Transform issues.

Similarly, Edge Detection algorithms, like Canny Edge Detection [7], were attempted and resulted unreliable due to the high encoding compression of the frames, blurring the edges of the surgical instruments. Another computer vision technique attempted to segment the instrument in the frames by leveraging its colour was Color Thresholding [7]. The surgical instrument, made of steel, is grey coloured and so it should have been easy to segment it based on its colour. Unfortunately, the light conditions inside the eyeball produce reflections on the steel body of the surgical tool, adding red, white, and yellow shades to the silver color of the tool. These reflections made Color Thresholding not applicable and highly unreliable in properly outlining the instrument's edges. As the instrument's tip resembles a very prominent feature in the frames, we also attempted some features detection & description algorithms among which ORB (Oriented FAST and Rotated BRIEF) [28] and SIFT (Scale Invariant Feature Transform) [29]. These algorithms look for features as distinctive corners. Edges are not good features as an edge-patch can usually match the entire edge while a corner-patch is unique. However, feature detection algorithms do not employ any knowledge about the task while detecting features. This pitfall, in our case, resulted in features identifying sharp blood vessels on the retinal surface or artefacts produced by noise and glares in addition to the instrument's edges, with no possibility to filter out the outliers. Unfortunately, there is no way to tell these algorithms to detect key points of the instrument's tip in place of other landmarks in the frame, such as prominent blood vessels. Examples of the mentioned issues are shown in Figure 17.

This last flaw led me to take into consideration the use of a supervised machine learning model as such a model can be trained to detect a feature of choice by providing a target. We

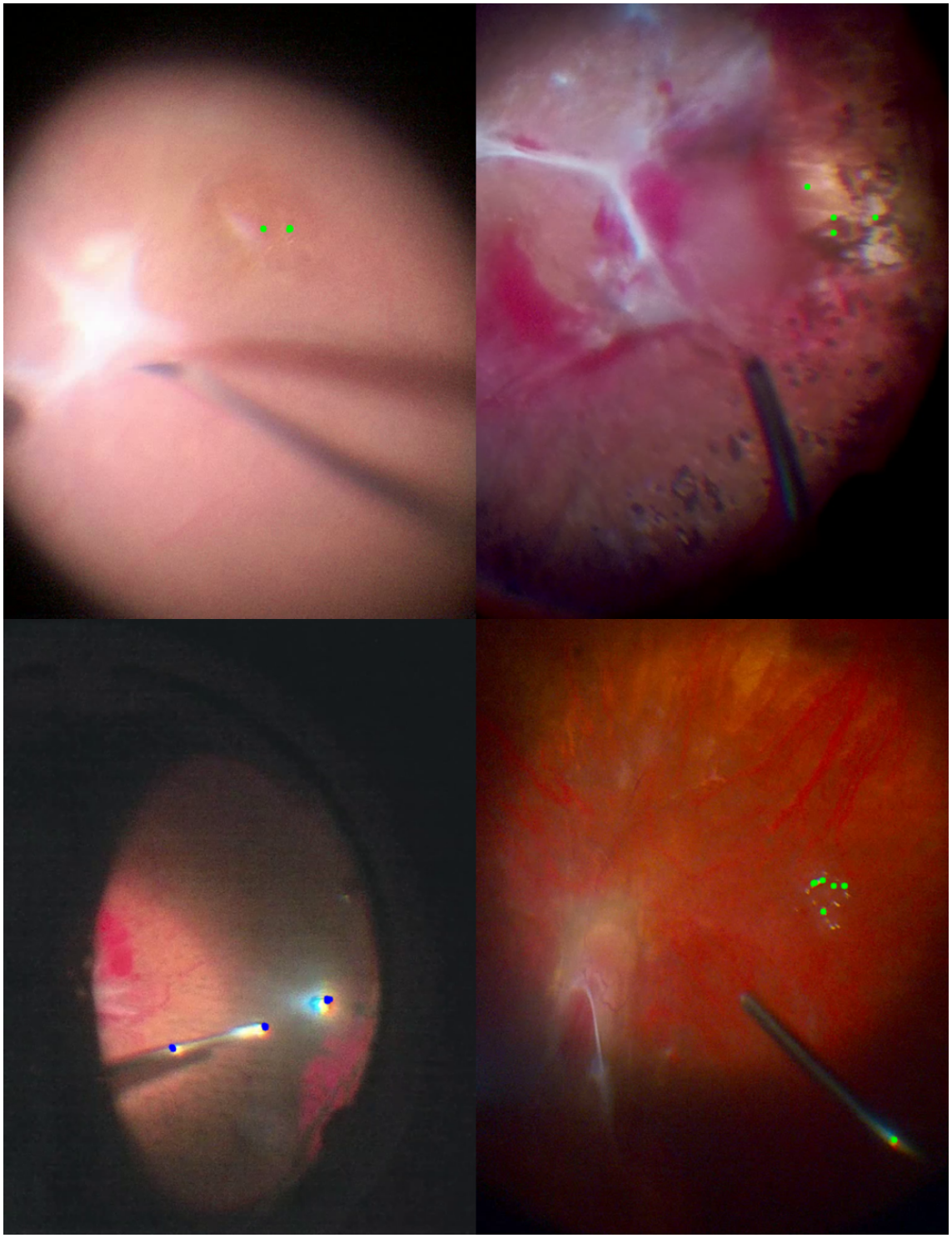

Figure 17: ORB issues.

eventually choose to employ a Convolutional Neural Network as CNNs are the current state-of-the-art approach for feature extraction and supervised tasks on visual data. We also attempted to leverage the movement of the surgical instrument to track it using Optical Flow [30]. Optical Flow is a Computer Vision that computes the movement vector of the pixels across subsequent frames. This movement vector allows having an understanding of what is moving in the frames, in which direction and so to track prominent pixels. We tried two different versions of the Optical Flow: a dense version, which computes the movement vector of every pixel in the frame, and a sparse version, which computes the movement vector for some few input points only. However, the dense version was very computational heavy and dramatically slowed down the entire pipeline when coupled with the stereo matching algorithm. Moreover, since the videos are not stabilized and the movement in the eyeball is random, the dense Optical Flow map presented a huge amount of noise, making the instrument's shape barely visible. A dense optical flow map with the mentioned issues can be seen in Figure 18.

The sparse Optical Flow was instead faster and free of noise. However, it needed some input points to track. These points should be as much accurate as possible to track the right features across frames. We already discussed the lack of accuracy of feature detection algorithms, like ORB and SIFT, making this class of algorithm unsuitable for our task. Our Convolutional Neural Network, on the other hand, was able to provide points that are accurate enough for the sparse Optical Flow. Eventually, we realized that it was faster and more reliable to locate the instrument tip in each frame through the CNN, instead of running the network once and tracking the located point with Optical Flow.

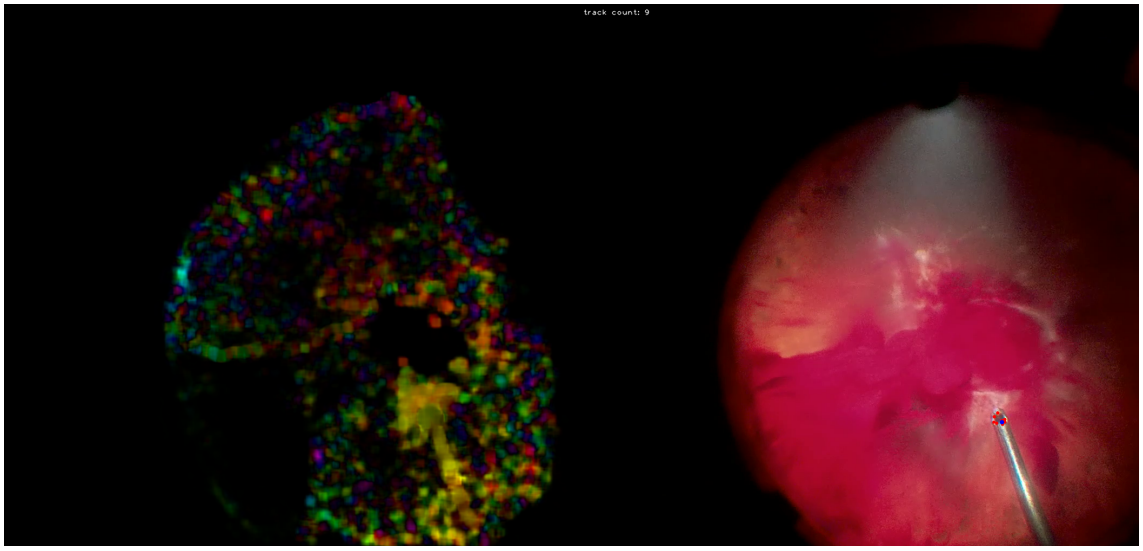

Figure 18: Dense Optical Flow issues.

The Artificial Neural Network we employed in our pipeline is a Convolutional Neural Network coupled with a Fully-Connected Feed-Forward Neural Network to perform Regression. The opening ConvNet extracts features from surgery frame; the Fully-Connected FFNN is then used to leverage those features and learn the coordinates of the instrument's tip in the frame. The topology of the network is inspired by [18], but in our case, it contains a total of six layers of 2D convolution, each of which with more filters than the CNN described in [18]. Moreover, our topology consists of a deeper and wider FFNN, with more layers and more neurons in each layer. Finally, we employ RGB images in place of gray images.

The input shape for the network is (320, 240, 3), representing RGB images with shape 320 x 240. The convolutional section is built with 6 2D-convolutional layers each followed by a

layer of Max Pooling and a layer of Dropout. The first convolutional layer has 32 kernels with size (3, 3), ReLU activation function and padding to do not lose any pixels during convolution. Starting from the second one, convolutional layers have (2, 2) kernels than double each time starting from 64 until 512. The activation function is still ReLU and padding is the same. All the max-pooling layers have a kernel size of (2, 2) and padding; dropout probability is 0.05 for the first layer, 0.1 for the second one and 0.2 for all the remaining layers. The Fully-Connected segment of our network consists of 4 dense hidden layers with 350 neurons each and ReLU activation function. The output layer has 2 neurons with ReLU. In Figure 19, Figure 20 and Figure 21 some features maps from different layers of the network for the input in Figure 22 are shown.

5.6 Adam Optimizer

In section 4.2.1 we stated that loss function optimization in Artificial Neural Networks is usually performed with Gradient Descend and in section 4.2 we cited that several implementations of this algorithm exist. Here we describe the one employed in our pipeline: Adam (Adaptive Moment Estimation) [9]. Adam is an adaptive learning rate optimizer that can update the value of the learning rate during the learning process based on the loss function gradient similarly to Adagrad [9], AdaDelta [9] and RMSprop [9]. Adam computes and stores both an average of the squared gradient with exponential decay and an average of the gradient again with exponential decay as

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{5.2}$$

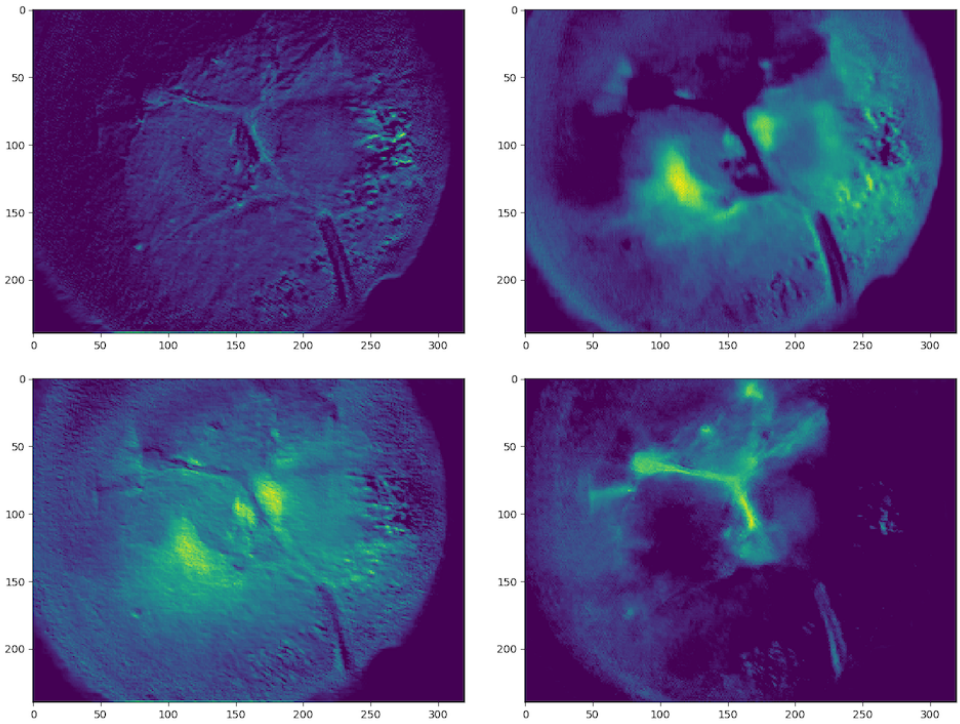

Figure 19: Some features maps from the first layer of our CNN.

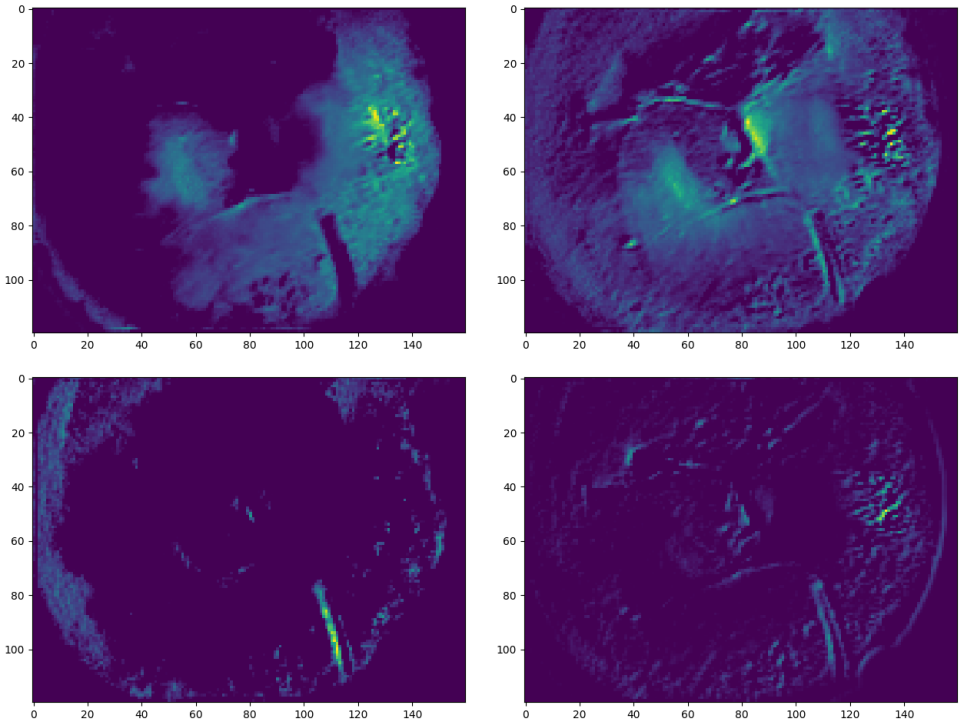

Figure 20: Some features maps from the second layer of our CNN.

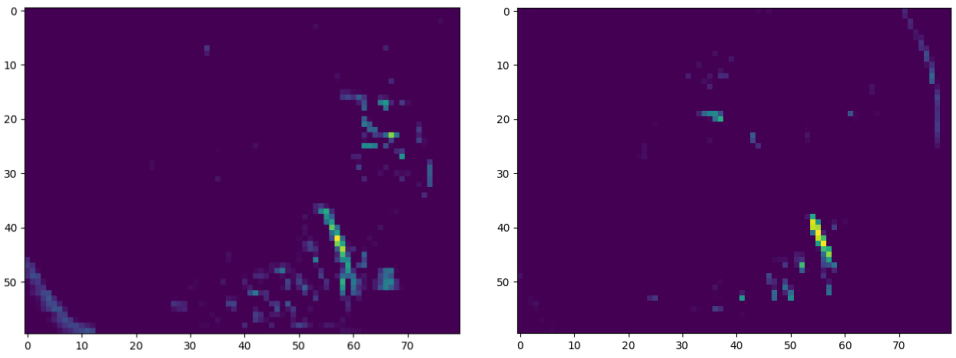

Figure 21: Some features maps from the third layer of our CNN.

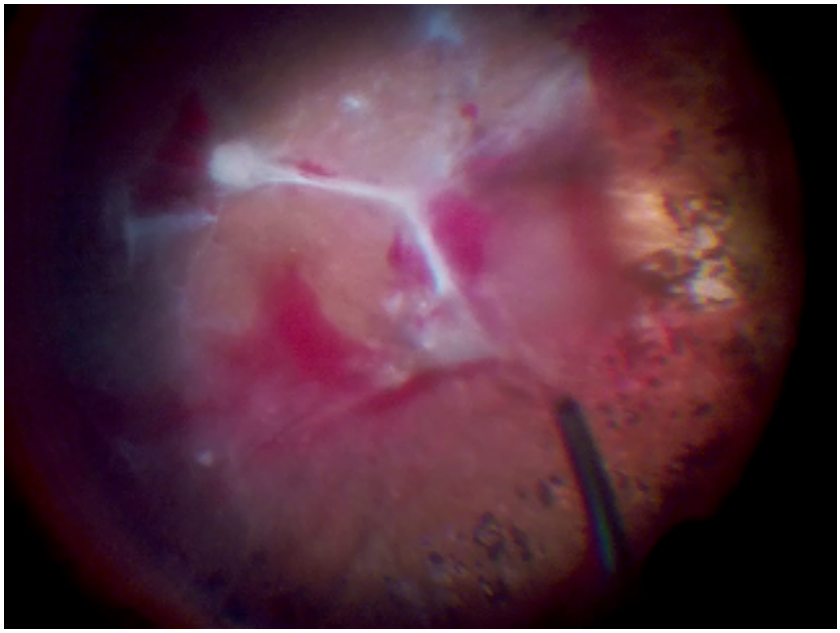

Figure 22: Input corresponding to the features maps above.

and

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{5.3}$$

where g_t is the loss function's gradient, m_t is the average of the gradient and v_t the average of the second gradient. Both these two estimates are biased towards zero as they are initialized to 0. For this reason, Adam employs unbiased versions of the above estimates as

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{5.4}$$

and

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{5.5}$$

This said, Adam update rule for parameters results to be

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{5.6}$$

where η is the learning rate, θ_t is the vector of the model's parameters at iteration t, and ϵ is worth 10^{-8}. We can notice the learning rate changing magnitude inversely proportionally to the unbiased estimate of the squared gradient.

5.7 Overfitting

In section 4.2.1 we mentioned the phenomenon of overfitting occurring in Artificial Neural Networks and some techniques to limit and solve it. Here we describe how we prevent overfitting to happen. The most basic expedient we used to avoid overfitting is dropout [9]. Dropout is a technique used in ANN to randomly disable connections between neurons and layers with probability p to improve generalization through this randomness. In Convolutional Neural Networks *dropout* is implemented by randomly zeroing weights inside the convolutional kernels. Dropout is applied during training but not during inference.

An additional technique we employed is Early Stopping [9]. *Early stopping* is a more advanced and widely used technique to limit overfitting that makes use of a validation set to check for overfitting during training. The validation set is a reserved part of the entire dataset of never seen samples employed to validate the generalization capabilities of the network on-line.

Network's performance on the validation set is compared to the performance on the training set to check for overfitting. Early Stopping stops training the network when the difference between the loss on the validation set and the training set begins to diverge. This difference means that the network is starting to overfit on training data, performing badly on the never seen samples of the validation set. The only parameter Early Stopping requires is the patience *i.e.*, the number of epochs to wait before stopping the training since the first divergence has been detected. Finally, we mention regularization in the form of Ridge regularization (L2) described in section 4.2.1 with magnitude 5×10^{-4} that we used on the first version of the CNN.

5.8 Typical Data

The raw data employed to carry out this research consists of videos of vitrectomy surgeries recorded using the surgical microscope present in the operating room. These videos, publicly available on the Internet, show the fundus and a surgical instrument manipuled by the ophthalmologist. The surgical tool is usually a straight pipe made of steel; however, some videos seldom show hooked tools and tongs. The scene is lit through a surgical torch also handled by the surgeon. The fundus is not always clearly visible as some scar tissue or blood clouds, which is usually the reason why the surgery is performed, can be presented between the microscope viewpoint and the fundus. The recorded surgery usually implies the surgeon chopping floating scar tissue, peel anchored scar tissue or remove blood clouds. The surgical microscope records the surgery both as standard, 2D video and as stereoscopic video. The latter version is the main data employed in this work. These videos are sampled to extract single frames, which compose the dataset for the Deep Learning algorithms. At this point, the dataset misses the

target variables, which in our case are the coordinates of the instrument's tip. To provide this information a marking tool was developed and employed. The marking tool allows the user to whether locate the instrument's tip in the frame and store its coordinates or identify the frame as invalid if no instrument is displayed in the frame. Sampling the videos, we had to choose a size for the images in the dataset. The perplexity was between using high-resolution images to preserve many details but enlarging the run-times or lower-resolution images to decreases the run-times. We then reached a tradeoff between information preserved and reduced run-time. No data augmentation was performed on the definitive version of the dataset. On the other hand, we performed undersampling by rebalancing a bias toward some patients to improve the generalization capabilities of our Convolutional Neural Network.

CHAPTER 6

EXPERIMENTAL SETUP

6.1 Implementation Details

The pipeline is entirely implemented in Python that has been preferred over other widely used programming languages such as C++ and Java because is considered to be the best language for prototyping due to its simple syntax. Moreover, most Deep Learning frameworks provide APIs for Python. For IO operations and image processing Python's OpenCV has been employed. OpenCV is an open-source computer vision library with plenty of built-in functions for image processing, IO operations, stereo vision, 2D features detection, segmentation and more. It provides APIs for C++, Python and Java. To deal with matrices and numerical computation NumPy has been employed, which is a library for optimized numerical computation and n-dimensional data representation. Pandas has been chosen to handle the data in the dataset through the Dataframe object. It is an open-source Python library for high-performance data structures. For Deep Learning programming, Keras with TensorFlow's backend has been used. TensorFlow is a Google's open-source library for numerical computation and machine learning with APIs for Python, Java, C++ and JavaScript; Keras is, again, an open-source Python library providing high-level APIs for neural networks training, test and deployment. It needs a computational backend, which we decided to be TensorFlow.

The current version of the pipeline is implemented to work with the recorded videos of ophthalmic surgeries described in section 5.8 but it can be easily modified to work with streams of data. We employed NumPy matrices indexing to crop the original frames from 1920×1080 to 1920×1026 to remove a logo on the bottom right corner. To center the region of interest (ROI) we utilized OpenCV's *threshold* function with the *Otsu* label to perform Otsu thresholding described in section 4.4.2. OpenCV's *findNotZero* function was then employed to find the left and right edges of the ROI and know where to crop. The outcome is an image with shape 1368×1026. This shape was chosen because multiple of 320×240, input shape of the Convolutional Neural Network employed in the pipeline. Resizing was performed with OpenCV's *resize* function. The Convolutional Neural Network was implemented with Keras and TensorFlow backend. All the details about it can be found in the next section. Histogram equalization has been performed with the OpenCV's build-in function *CLAHE* with contrast limited to 1.0 and mask size of (3, 3). Disparity maps were computed with images 1920×1026 with OpenCV's *stereoSGBM*. The parameters employed for SGBM were adapted to each video as we figure out that it was not possible to find a unique set of parameters producing good disparity maps.

6.2 CNN's Training Environment

As mentioned in section 6.1, a Convolutional Neural Network has been employed in our pipeline to estimate the coordinates of the instrument's tip in the frames. We described the version and the structure of our Convolutional Neural Network in section 5.5. However, some additional details are needed to fully depict the CNN used.

We decided to employ RGB images instead of greyscale (1-channel) images to preserve the majority of the details in the images, persuaded that a certain amount of useful information is contained in colours. We used a batch size *i.e.*, the number of samples considered at each step of the optimization, equal to 64. Learning rate was chosen with an initial value of 0.001 (10^{-3}) but with the possibility to dynamically change it with Adam optimizer. Optimizers are the implementations of Gradient Descent algorithms. We choose Adam as it is well known to be faster to converge on large dataset and reliable.

We split our dataset as 90% training set, 10% validation set and 10% test set. We deployed Early Stopping, described in section 5.7, with patience 15 epochs and Model Checkpoint with frequency 1, which stores the values of the weights each time a new minimum in the loss function is reached. This technique ensures to always have the best set of weights so far. In section 5.5, we stated that we configured the two output neurons of the network with ReLU (Rectified Linear Unit) activation function. Here we explain why: since we are performing a regression task on pixels coordinates, we want neurons that can output non-negative real values, as pixels' coordinates range from 0 to *image-width* and from 0 to *image-height*. In the final version of the network depicted in section 5.5, we did not add regularization as the number of samples in the dataset resulted enough. Moreover, regularization prevented the network to fit the data properly resulting in bad performances. We defined a custom, still widely used, loss function implementing the Euclidean Distance,

$$\sqrt{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2} \tag{6.1}$$

where y_i is the produced output, $\hat{y}_i$ is the expected value, and n is the number of samples. We employed this loss function as the error we want to minimize in the network is the distance between the predicted point and the ground truth.

6.3 Training Dataset

Data is a core aspect of Machine Learning, as already said in section 4.2 and 4.3. We were provided with videos recorded by two 4K stereo cameras at 60 FPS but provided to the user with resolution 1920×1080 (FullHD) in the monocular version and 3840×1080 in the stereoscopic version. However, to train our CNN we needed a dataset containing surgery images and instrument tip coordinates as the target. No such dataset is available on the Internet so we had to build our dataset from scratch with the frames of the surgeries videos.

The first step was to choose a size for the images in the dataset. The trade-off was between loss of too many details with a very low resolution and long computational run-times with a high resolution. Eventually, we opted for the shape 320×240, which preserves the majority of details but drastically decreases the run-time when training and deploying the CNN. We then sampled frames from 7 distinct videos at 1 frame every 10. Subsampling was required as the provided videos have a high frame rate, causing subsequent frames to be very similar to each other, which is bad if we want to train a CNN. This way we produced a non-labelled dataset containing nearly 10 thousand 320×240 images of 7 different surgeries. Some frames are shown in Figure 23. The next step was to annotate all the frames with the coordinates of the instrument tip if present.

To obtain these annotations, we built a custom labelling tool using TkInter, a Python library for GUI (Graphical User Interface). The tool allows the person in charge of providing the frames' annotations to point at the instrument's tip in the frames and save its coordinates in a CSV file. In case of instrument missing, tip occluded or frame corrupted, the user is allowed to invalidate the frames through a button in the GUI. This custom tool allowed us to annotate the 10k frames by hand in about 20 hours of work. In the end, 9.5% of the frames resulted invalid and 90.5% resulted valid *i.e.*, suitable for the CNN.

Two versions of the dataset were built and employed for the CNN training. The first version consisted of annotated frames sampled from the videos plus additional frames created with data augmentation. Data augmentation is an oversampling technique used to increase the number of samples in the dataset by applying transformations to the samples in the dataset. Examples of possible transformations are rotation, flip, zoom-in, zoom-out, pixels intensities variation and noise addition. However, on this first version, our CNN was not performing at the best of its capabilities. This version contained in fact two main issues: a bias introduced by the strong data augmentation performed and an intrinsic bias towards surgeries with more frames in the dataset. So, instead of performing hyper-parameters tuning and changes in the network topology, we decided to extensively work on the data to improve its quality. In brief, we substituted oversampling via data augmentation with under-sampling that, as underlined in several papers among which [31], usually works better. The second and current version of the dataset was the result of this process: we completely dropped data augmentation that it is

not strictly needed but can be added in the future and we balanced the number of frames from each surgery, so that to remove any bias towards a particular scenario.

We also performed a little bit of data exploration to deeply understand our data before feeding the our CNN. We plotted the distribution of the targets along the x and y-axis discovering that the x-components have a Gaussian distribution with the mean around the centre of the image and the y-components have a bi-modal distribution. These distributions explain why the CNN outputs some coordinates near the centre of the image when no tip is shown in the frame.

For testing purposes, we kept aside two videos that the network never saw during the training so that we tested the performance of the pipeline on never-seen images.

6.4 Testing Environment

In this section, we describe the hardware and software environments used to develop our proposed pipeline. The majority of the pipeline was developed using a laptop running macOS 10.14.6 with:

- a 2.6 GHz dual-core Intel i5 CPU;
- an Intel Iris integrated GPU;
- 8 GB DDR3 of RAM at 1600 MHz;
- 128 GB SSD of storage.

This machine was used to run all the tests regarding image processing. However, some sections of the pipeline are too computational heavy for the machine described above. For this reason,

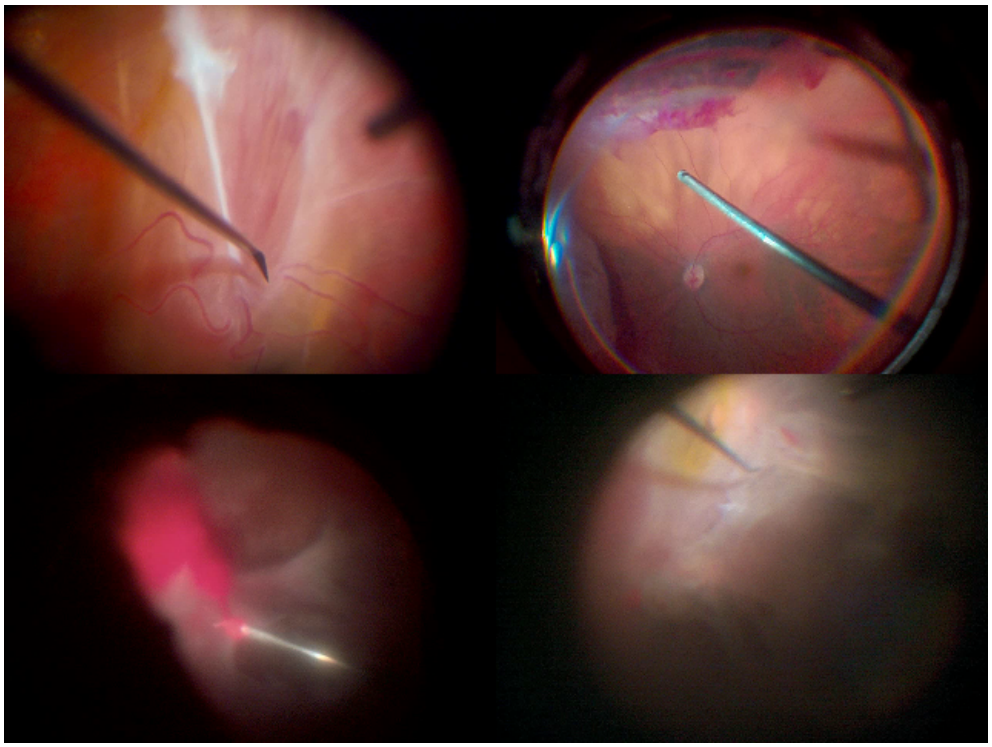

Figure 23: Some frames from the dataset.

a workstation has been used to train the Convolutional Neural Network and test the entire pipeline. This machine runs Windows 10 with:

- a 3.7 GHz exa-core Intel i7 CPU;
- an Intel UHD Graphics 630 integrated GPU;
- 16 GB of RAM at 2666 MHz;
- 256 GB SSD of system storage;
- 1TB HDD of additional storage;
- 2x NVIDIA GeForce GTX 1070 Ti with 16GB dedicated GPU.

On both machines, we employed JetBrains' PyCharm IDE. However, on the laptop, we employed a Python virtual environment with standard Tensorflow installed while on the workstation a Python virtual environment with Tensorflow-GPU to take advantage of the computational power of the GPUs.

TABLE I: LAPTOP'S SPECIFICATIONS SUMMARY.

CPU	RAM	Integrated GPU	Dedicated GPU	Storage
2.6 GHz dual-core Intel i5	8 GB DDR3 1600 MHz	Intel Iris	n/a	128 GB SSD

TABLE II: WORKSTATION'S SPECIFICATIONS SUMMARY.

CPU	RAM	Integrated GPU	Dedicated GPU	Storage
3.7 GHz exa-core Intel i7	16 GB DDR4 2666 MHz	Intel UHD Graphics 630	2x NVIDIA GeForce GTX 1070 Ti 16GB	256 GB SSD + 1TB HDD

CHAPTER 7

RESULTS

In this chapter, we present the results achieved by our pipeline. Since the general performance of the pipeline depends on how our CNN performs, we first present the training and test performances of our network.

7.1 CNN's Training Metrics

We already mentioned in section 6.4 that we trained our Convolutional Neural Network on a workstation with two NVIDIA GeForce GTX 1070 Ti. The final model deployed in our pipeline took 28 minutes on the workstation to train; the training loss reached a value of 0.0120 (12×10^{-3}) and the validation loss 0.0132 (13.2×10^{-3}) before the Early Stopping stopped the training process after 74 epochs, as showed in Figure 24.

The loss function on the test set with the weights stored in the model checkpoint measured 0.015 (15×10^{-3}). We plotted the distribution of the errors over the test set containing 657 samples that can be seen in Figure 25.

The mean error resulted to be 3.24 pixels with a standard deviation of 4.17. The max error performed was of 87.81 pixels and the minimum 0.11 but the interesting statistics is that the 90th-percentile is 5.53 pixels. This value means that 90% of the samples in the test set had an error less than or equal to 5.53 pixels.

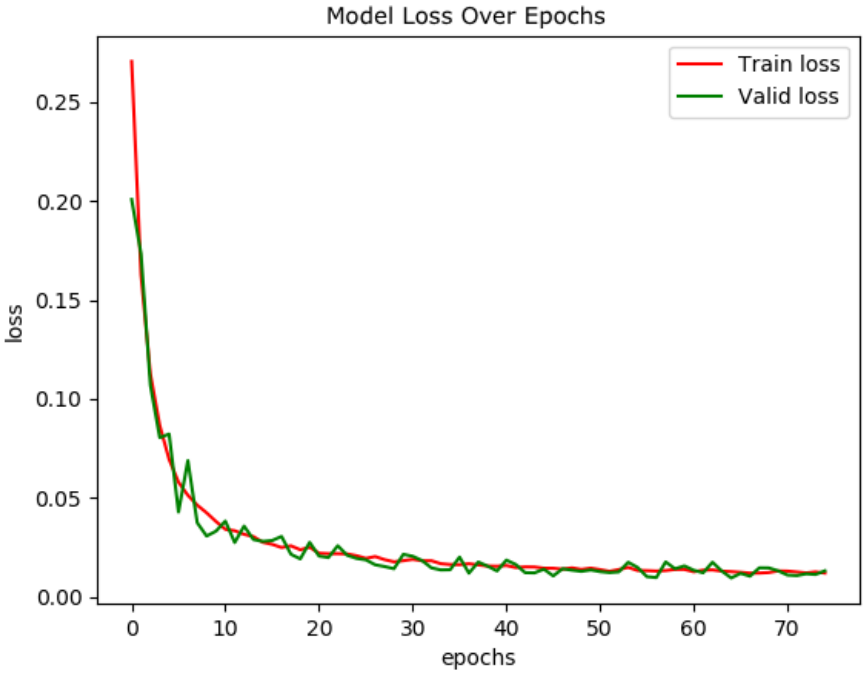

Figure 24: The loss function plot on training and validation set.

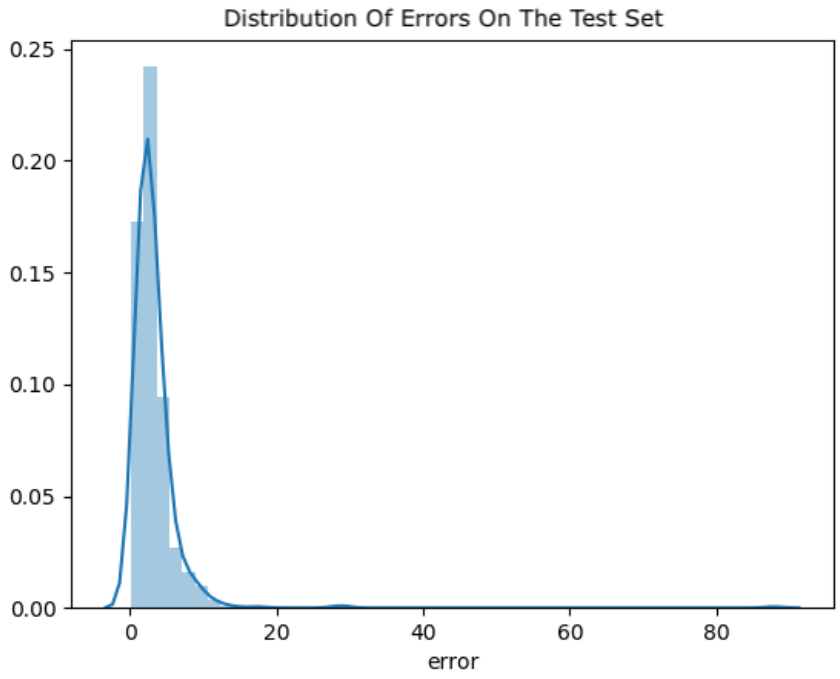

Figure 25: The distribution of errors performed by our CNN on the test set.

We also tested the final performance of our CNN on videos as this is the environment it has to be deployed in. We tested our network on both videos we used to extract frames as we sampled images 1 every 10 and those videos still contain a large number of new frames, and never-seen-before videos we excluded from the dataset.

TABLE III: PERFORMANCE OF OUR CNN ON THE TEST SET.

Count	Mean	Std Dev	Min	25%	50%	75%	90%	Max
657	3.24	4.17	0.11	1.72	2.57	3.78	5.53	87.81

In Figure 26 we show some inferences of our Convolutional Neural Network on never-seen images. Based on the output of the network, we drew a box, which is more meaningful as the pipeline employs a pixel mask and not a single point.

7.2 Pipeline Performance

We already discussed the accuracy of the Convolutional Neural Network in the previous section. About the depth estimation, the value produced by the pipeline seems to be consistent with the scene during the majority of the video. However, due to a series of factors among which lack of rectification and lack of texture in the frames, some false negatives and false positives are produced by the pipeline. A false positive means that the pipeline issues a warning even

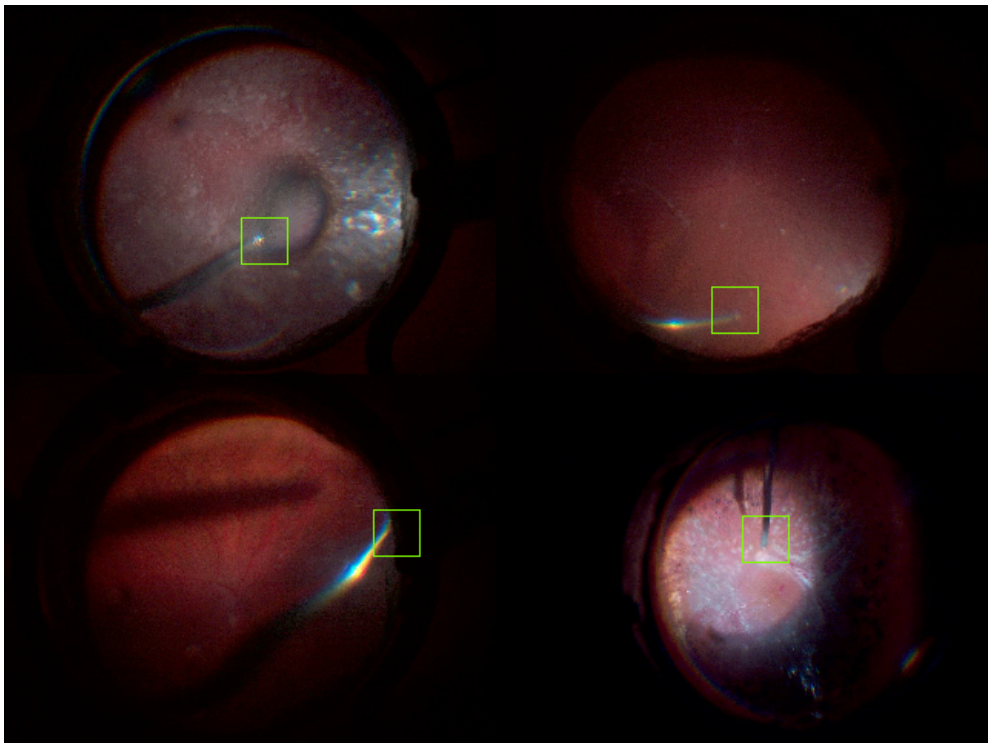

Figure 26: Samples of inferences of our network on never-seen frames.

if the instrument tip is not dangerously close to the retina; a false negative means that the pipeline does not issue a warning when needed.

Concerning the runtime, a single iteration of the pipeline takes between 70 and 100 milliseconds. More in details, our Convolutional Neural Network takes about 2-3 milliseconds to produce an output, the Semi-Global Block Matching algorithm takes between 30 and 65 milliseconds to compute the disparity map and the remaining time is due to IO operations and helper functions, like cropping, resizing, histogram equalization and thresholding. As it can be noticed, the computation of the disparity maps is the performance bottleneck. It may be speeded up by employing lower resolution images for stereo matching but we would be concerned about the loss of details.

Figure 27 and Figure 28 show the pipeline running on a video with the GUI highlighting the position of the instrument tip and indicating the current frame. The distance warning is also displayed in Figure 27.

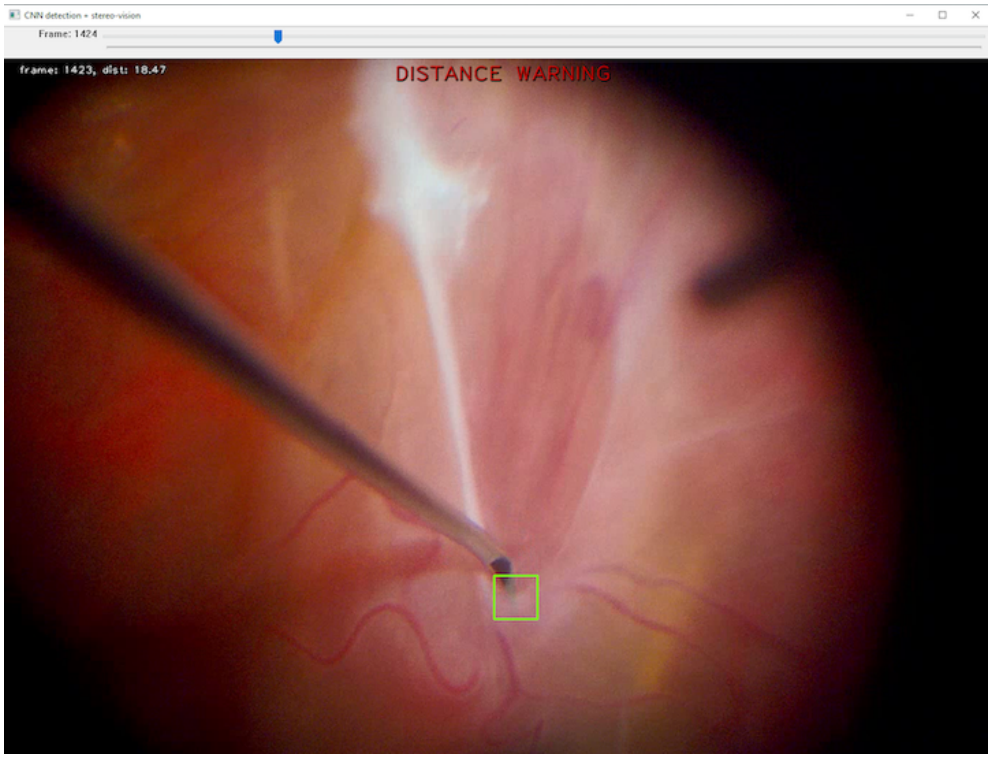

Figure 27: Screenshot of the pipeline deployed - part 1.

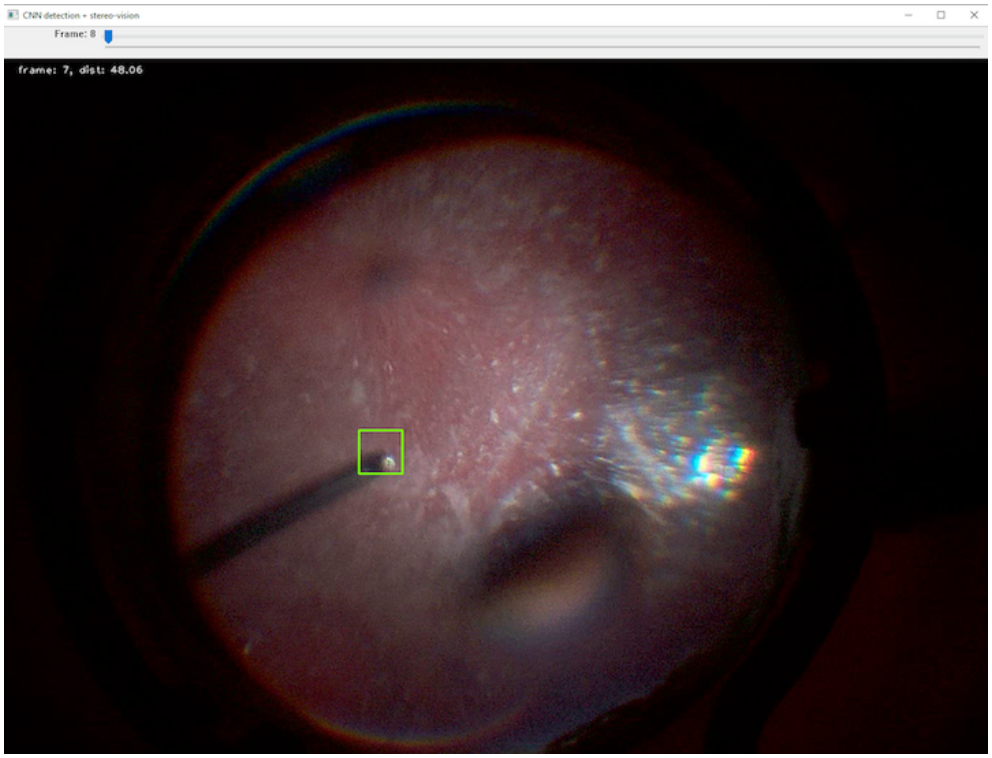

Figure 28: Screenshot of the pipeline deployed - part 2.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusion

In conclusion, we designed and developed an automated, real-time pipeline, which represents a proof of concept for the employment of Convolutional Neural Networks and Stereo Vision to estimate the relative distance of the instrument tip from the retina surface.

Overall, the performance of the Convolutional Neural Network in inferring the location of the instrument tip in the frames can be considered to be acceptable for real-time visualization as shown by the statistics in section 7.1. Of course, CNN's inferring capability can be enhanced with additional data and with further effort on its topology. The quality of the disparity maps employed in the pipeline is not always satisfactory, causing some unwanted false positive and false negative. As discussed in section 4.4.1.2, we should have performed rectification before applying stereo matching and computing disparity maps as stereo matching algorithms are very sensitive to distortion. However, we could not rectify the images as we lack intrinsic parameters and transformations H1 and H1 described in section 4.4.1.2. Moreover, the quality of the disparity maps shows a large variance across different videos due to the lack of generalization of the parameters of Semi-Global Stereo Matching. This variance means that a set of optimal parameters and fine-tuning is needed for each video to always obtain good disparity maps.

Concerning the entire pipeline, we think that its accuracy in tracking the instrument tip and estimating its depth is satisfactory for the purposes of this thesis. About real-time performance, the pipeline runs in real-time at 24 frames per second on mid-range hardware, being easily deployable in real microsurgical settings.

8.2 Future Work

Given the results and the limitations of our proposed pipeline described across Chapter 7 and section 8.1, we foresee a great variety of future extensions to this work. First of all, achieving image rectification in this scenario could lead to a terrific improvement in the quality of disparity maps and of the entire pipeline as consequence as discussed in section 4.4.1.2. In general, a better and more stable quality for the disparity maps is desirable to improve the performance of the pipeline. In this sense, the employment of ad-hoc Convolutional Neural Networks for the computation of disparity maps represents a viable path. CNNs could, in fact, overcome the lack of generalization of SGBM mentioned in section 8.1.

Moreover, some additional efforts can be spent on the dataset and the network structure to improve the inference accuracy of our CNN. For instance, in section 6.3 we stated our choice of dealing with 320 x 240 images as the real-time performance was a concern: modify the dataset and the structure of the CNN to deal with higher resolution images is an option to improve inference quality. Other viable alternatives are expanding the dataset or carefully analyze the activation maps of the hidden layers of the network to drive modifications in our CNN's structure.

In conclusion, Recurrent Neural Networks (RNN) represent another possible extension. As mentioned in section 4.2.1, RNNs are Artificial Neural Networks that can work on data with some kind of time dependency. They leverage not only spatial features but also temporal features. As there exists a strong time correlation between frames across surgery videos, RNNs can be successfully employed to deal with videos as a stream of data and not only as a sequence of independent frames to enhance the performance of the pipeline. Our only concern about this approach may be represented by real-time performance of RNNs, as they are computationally heavy.

CITED LITERATURE

1. Wikipedia - Human Eye. https://en.wikipedia.org/wiki/Human_eye. [Online; Last access: 11/26/2019].

2. ASRS - Vitrectomy. https://www.asrs.org/patients/retinal-diseases/25/vitrectomy. [Online; Last access: 11/20/2019].

3. Personal communication with Dr. Yannek I. Leiderman.

4. Truevision - Leica Microsystems. https://www.leica-microsystems.com/products/surgical-microscopes/details/product/truevision-integrated-3d/. [Online; Last access: 11/20/2019].

5. American Association of Ophthalmology - OCT. https://www.aao.org/eye-health/treatments/what-is-optical-coherence-tomography/. [Online; Last access: 11/20/2019].

6. De Silvestri, M.: Real-time Haptic Guidance System for Retinal Surgery Based on Intraoperative Optical Coherence Tomography, 2018.

7. Szeliski, R.: Computer Vision: Algorithms and Applications. Berlin, Heidelberg, Springer-Verlag, 1st edition, 2010.

8. Wikipedia - Computer Vision. https://en.wikipedia.org/wiki/Computer_vision. [Online; Last access: 11/26/2019].

9. Goodfellow, I., Bengio, Y., and Courville, A.: Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org. [Online; Last access: 12/03/2019].

10. Mitchell, T. M.: Machine Learning. New York, NY, USA, McGraw-Hill, Inc., 1 edition, 1997.

11. Bishop, C. M.: Pattern Recognition and Machine Learning. Berlin, Heidelberg, Springer-Verlag, 2006.

CITED LITERATURE (continued)

12. Ramaswamy, A., Ram, K., and Sivaprakasam, M.: A depth based approach to glaucoma detection using retinal fundus images. 2016.

13. Hecker, S., Probst, T., Chhatkuli, A., Maninis, K.-K., Havlena, M., Vander Poorten, E., and Van Gool, L.: Online reconstruction of retinal 3d map from stereo in close-to-realtime for eye surgery. In Proceedings of the 7th Joint Workshop on New Technologies for Computer/Robot Assisted Surgery, pages 34–35. KU Leuven, 2017.

14. Lam, C., Yu, C., Huang, L., and Rubin, D.: Retinal lesion detection with deep learning using image patches. Investigative ophthalmology & visual science, 59(1):590–596, 2018.

15. Shankaranarayana, S. M., Ram, K., Mitra, K., and Sivaprakasam, M.: Fully convolutional networks for monocular retinal depth estimation and optic disc-cup segmentation. IEEE journal of biomedical and health informatics, 2019.

16. Van Grinsven, M. J., van Ginneken, B., Hoyng, C. B., Theelen, T., and Sánchez, C. I.: Fast convolutional neural network training using selective data sampling: Application to hemorrhage detection in color fundus images. IEEE transactions on medical imaging, 35(5):1273–1284, 2016.

17. Gao, X., Lin, S., and Wong, T. Y.: Automatic feature learning to grade nuclear cataracts based on deep learning. IEEE Transactions on Biomedical Engineering, 62(11):2693–2701, 2015.

18. Al-Bander, B., Al-Nuaimy, W., Al-Taee, M. A., Al-Ataby, A., and Zheng, Y.: Automatic feature learning method for detection of retinal landmarks. In 2016 9th International Conference on Developments in eSystems Engineering (DeSE), pages 13–18. IEEE, 2016.

19. Menegola, A., Fornaciali, M., Pires, R., Avila, S., and Valle, E.: Towards automated melanoma screening: Exploring transfer learning schemes. arXiv preprint arXiv:1609.01228, 2016.

20. Smits, J., Ourak, M., Gijbels, A., Esteveny, L., Borghesan, G., Schoevaerdts, L., Willekens, K., Stalmans, P., Lankenau, E., Schulz-Hildebrandt, H., et al.: Development and experimental validation of a combined fbg force and oct distance sensing needle for robot-assisted retinal vein cannulation. In

CITED LITERATURE (continued)

2018 IEEE international conference on robotics and automation (ICRA), pages 129–134. IEEE, 2018.

21. Wikipedia - RMSE. https://en.wikipedia.org/wiki/Root-mean-square_deviation. [Online; Last access: 11/25/2019].

22. Wikipedia - RSS. https://en.wikipedia.org/wiki/Residual_sum_of_squares. [Online; Last access: 11/25/2019].

23. Wikipedia - R2. https://en.wikipedia.org/wiki/Coefficient_of_determination. [Online; Last access: 11/25/2019].

24. Jain, R., Kasturi, R., and Schunck, B. G.: Machine Vision. New York, NY, USA, McGraw-Hill, Inc., 1995.

25. Thresholding. https://docs.opencv.org/4.2.0/d7/d4d/tutorial_py_thresholding.html. [Online; Last access: 11/25/2019].

26. Wikipedia - Otsu's Method. https://en.wikipedia.org/wiki/Otsu%27s_method. [Online; Last access: 11/25/2019].

27. Wikipedia - Semi Global Matching. https://en.wikipedia.org/wiki/Semi-global_matching. [Online; Last access: 11/25/2019].

28. Rublee, E., Rabaud, V., Konolige, K., and Bradski, G.: Orb: An efficient alternative to sift or surf. In Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

29. Lowe, D. G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision, 60(2):91–110, November 2004.

30. Wikipedia - Optical Flow. https://en.wikipedia.org/wiki/Optical_flow. [Online; Last access: 12/01/2019].

31. Li, S., Zhou, G., Wang, Z., Lee, S. Y. M., and Wang, R.: Imbalanced sentiment classification. In Proceedings of the 20th ACM international conference on Information and knowledge management, pages 2469–2472. ACM, 2011.

VITA

NAME	Mattia Di Fatta
EDUCATION	
May 2020	M.S. in Computer Engineering, University of Illinois at Chicago, USA
April 2020	M.S. in Computer Science & Engineering, Politecnico di Milano, Italy
July 2017	B.S. in Computer Science & Engineering, Politecnico di Milano, Italy
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	A.Y. 2017/2019 Lessons and exams attended exclusively in English
	2018 - TOEFL iBT examination (96/120)
	A.Y. 2018/19 One Year of study abroad in Chicago, Illinois
SCHOLARSHIPS	
N/A	N/A
TECHNICAL SKILLS	
Above average	Programming (Python, Java, C, JavaScript, Android)
Above average	Machine learning
Above average	Data Mining
Above average	Deep Learning
Above average	Web Development
PROJECTS	
Fall 2018	**Proteins Detection In Sub-Cellular Structures with CNNs** Computational Biology class project.
Spring 2019	**Online Bookstore** Hypermedia applications class project.
Spring 2018	**Sales Forecast** Data & Text Mining class project.
Spring 2018	**Hand tracking on low-end hardware with CNNs** Deep Learning class project class project.