

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Visual Odometer on Videos of Endoscopic Capsules (VOVEC)

Gil Martins Pinheiro

MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Supervisor: António Manuel Trigueiros de Silva Cunha

Second Supervisor: Hélder Filipe Pinto de Oliveira

June 25, 2018

Visual Odometer on Videos of Endoscopic Capsules (VOVEC)

Gil Martins Pinheiro

MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE
COMPUTADORES

June 25, 2018

Resumo

A cápsula endoscópica é um dispositivo do tamanho de um comprimido vitamínico que é engolido pelo paciente, gravando 8 a 10 horas de vídeo do seu trato gastrointestinal. Desde a sua introdução em 2001, tornou-se o principal método para diagnosticar doenças no intestino delgado, uma região de difícil acesso para métodos endoscópicos tradicionais. No entanto, estas cápsulas não são capazes de fornecer qualquer informação de localização, mesmo sendo esta vital para o diagnóstico e acompanhamento de doenças e para intervenções cirúrgicas no intestino delgado. Atualmente, a localização é estimada através dos escassos marcos existentes no tubo digestivo ou por métodos que recorrem a *hardware* externo que causam desconforto nos pacientes e tornam o processo mais dispendioso. Os métodos existentes baseados em software mostram grande potencial, mas ainda apresentam algumas limitações.

Esta dissertação tem como objetivo contribuir para melhorar a localização da cápsula no intestino delgado através da utilização de odometria visual e assim ir ao encontro das necessidades dos clínicos da área. Para isso, primeiro são selecionadas as imagens informativas, i.e. em que a aproximação da homografia é possível, depois são estimadas as matrizes homografia entre pares de imagens consecutivas e, finalmente, são estimados os deslocamentos em percentagem. Para o cálculo da homografia foi utilizado um método supervisionado de *deep learning* adaptado para os desafios das imagens de vídeos endoscópicos.

O método foi treinado num *dataset* com 7303680 amostras de treino com *ground truths* gerados sinteticamente, sendo que as matrizes homografia estimadas obtiveram um erro de 1.36 píxeis num *dataset* de teste com 2711040 amostras. A estimação do deslocamento foi comparada com os resultados obtidos no *Rapid Reader* e resultou em erros médios absolutos de $5.00 \pm 2.65\%$ e $3.16 \pm 2.08\%$, valores que evidenciam o potencial do método para utilização em ambiente clínico.

Adicionalmente, experimentamos com um método não supervisionado de *deep learning* que permite a estimação da posição tridimensional da cápsula endoscópica ao longo do intestino delgado.

Abstract

Endoscopic capsules are vitamin-sized devices that leverage from a small wireless camera to create 8 to 10 hour videos of the patients digestive tract. Since their introduction in 2001, they have become the leading diagnosing method for the small bowel, a region not easily accessible with traditional endoscopy techniques. However, the capsules do not provide localization information, despite being crucial for the diagnosis, follow-ups and surgical interventions on the small intestine. Currently, the capsule localization is either estimated based on scarce gastrointestinal track landmarks or given by additional external hardware that cause further discomfort on the patient and represent a cost increase. Current software methods show great potential, but still need to improve in order to overcome their limitations.

This dissertation will focus on improving the endoscopic capsule localization inside the small bowel through visual odometry, always considering the actual needs of gastroenterologists. In this regard, we first select the informative images, i.e. where the homography approximation is possible, then estimate the homography matrices between consecutive frame pairs and, finally, compute the displacement between each frame in percentage. In order to estimate the homographies, a supervised deep learning method was adapted for use with the challenging endoscopic images.

The method was trained in a dataset with 7303680 samples with synthetically generated ground truths, obtaining homography matrices approximations with an error of 1.36 pixels in a test dataset with 2711040 samples. The displacement computation was compared with the results obtained in *Rapid Reader*, resulting in mean absolute errors of $5.00 \pm 2.65\%$ and $3.16 \pm 2.08\%$, values that highlight the method potential for clinical implementation.

Additionally, we experimented with an unsupervised learning method that allows us to estimate the capsule 3D pose along the small bowel in an end-to-end manner.

Acknowledgements

I would like to express my sincere gratitude to my supervisors António Manuel Trigueiros de Silva Cunha and Hélder Filipe Pinto de Oliveira for their patience, guidance, knowledge and for introducing me to this topic.

Furthermore, I am very grateful for the opportunity to collaborate with Centro Hospitalar do Porto (CHP), especially with doctor Marta Salgado.

I would like to extend my sincere thank to INESC Porto, more specifically to the center for Biomedical Engineering Research (C-BER) for all the work conditions, infrastructure and technical support given.

I would also like to thank Paulo Coelho for his contributions in term of topographic annotations.

On a more personal note I would like to thank my family, more specifically my parents and brother. Without them this would not be possible, thank you for giving me all the love and education I needed. Also, a very special thanks to my girlfriend for her patience, love, company and for inspiring me to work hard and be my better self every day. Last but not least, I would like to compliment my friends for their friendship and support, specially for those who accompanied me throughout my academic journey. They also helped me grow and become a better person.

Gil Martins Pinheiro

*“Study hard what interests you the most in the most undisciplined,
irreverent and original manner possible.”*

Richard Philips Feynman

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	2
1.3	Contributions	3
1.4	Document Structure	3
2	Literature Review	5
2.1	Multiple View Geometry	5
2.1.1	Classical Techniques	5
2.1.1.1	Camera Model	6
2.1.1.2	Structure from motion	9
2.1.1.3	Local Features	11
2.1.1.4	Classical Homography Estimation	21
2.1.2	Deep Learning Approach	25
2.1.2.1	Supervised Learning	26
2.1.2.2	Unsupervised Learning	27
2.1.2.3	Convolutional Neural Networks	28
2.1.2.4	Deep Homography Estimation	30
2.1.2.5	Depth and Ego-Motion Estimation	35
2.2	Endoscopic Capsules Position Estimation	40
2.2.1	Capsule Localization Methods Overview	41
3	Visual Odometry Framework	43
3.1	Problem Characterization	43
3.2	System Overview	44
3.2.1	Displacement Estimation - HomographyNet	45
3.2.2	Depth and Pose Estimation - SfMLearner	45
3.3	Dataset	46
3.3.1	Treated Dataset	48
3.3.2	Camera Calibration	49
4	Results	51
4.1	Displacement Estimation - HomographyNet	51
4.1.1	Network Architectures	51
4.1.2	Synthetic Dataset	53
4.1.3	Train and Synthetic Data Tests	55
4.1.4	Real Data Tests	57
4.1.5	Displacement Computation	61

4.2	Depth and Pose Estimation - SfMLearner	67
4.2.1	SfMLearner Dataset	67
4.2.2	Training	68
4.2.3	3D Pose and Depth Results	69
5	Conclusions and Future Work	71
5.1	Objectives Satisfaction	71
5.2	Future Work	73
	References	75

List of Figures

2.1	General progress for 3D reconstruction	7
2.2	Pinhole camera model	8
2.3	Rearranged pinhole camera model	8
2.4	Simplified camera intrinsics	9
2.5	Two-view geometry based on the pinhole camera model	10
2.6	Features from a contour image, on the left, and a grayscale image, on the right, respectively	11
2.7	Corner detection	14
2.8	SIFT keypoints	14
2.9	SIFT feature descriptor creation	16
2.10	Disparity calculation	17
2.11	Results from the rectification algorithm presented by Loop and Zhang	18
2.12	Results from the rectification algorithm presented by Isgrò and Trucco	19
2.13	Feature matching between two images	20
2.14	Generic feature-based algorithm	23
2.15	One-to-one mapping of the 4-point homography matrix to the 3x3 homography matrix	24
2.16	Forward and backpropagation in neural networks with multiple layers	27
2.17	LeNet-5, an example of a convolutional neural network	28
2.18	Example of filters learned by the first convolutional layer of ImageNet	29
2.19	Max pooling operation example	30
2.20	HomographyNet	31
2.21	Corner Confidences Measure as a result of the <i>Classification HomographyNet</i>	31
2.22	Depth-wise and point-wise convolutional filters	32
2.23	Homography estimation MobileNet-based architecture head	33
2.24	General view of the homography estimation unsupervised model	35
2.25	General SfMLearner training pipeline based on view synthesis	36
2.26	SfMLearner training pipeline	37
2.27	Differentiable warping process	38
2.28	Explainability masks examples where highlighted pixels are branded unexplainable	39
2.29	SfMLearner architectures	40
2.30	<i>Given Imaging PillCam Reader Software v9</i> showing the small bowel map, progress bar and thumbnails	42
3.1	System framework overview	44
3.2	Displacement estimation framework	45
3.3	3D pose and depth map estimation framework	46
3.4	PillCam SB2 and SB3 examples	47

3.5	PillCam SB3 consecutive frames without superimposition.	48
3.6	PillCam SB3 captured frames with noise. We consider noise the green residues and bubbles that occlude the features.	48
3.7	<i>MATLAB Single Camera Calibrator App</i> framework	49
4.1	Original and perturbed patches	55
4.2	Data augmentation	55
4.3	Graphical representation of the regression HomographyNet results. From left to right we have represented the <i>Euclidean L2</i> loss and the <i>Mean Corner Error (MCE)</i> metric in pixels. In blue we have the results in the training set and in orange in the test set.	56
4.4	Graphical representation of the MobileNet-based HomographyNet results	56
4.5	Multiple examples showing the predictions accuracy	57
4.6	Comparison between predictions with distorted and undistorted images	58
4.7	Comparison between three alternative homography estimation approaches	59
4.8	Multiple examples showing the homography predictions in real data	60
4.9	Unsuccessful homography prediction between two consecutive frames	61
4.10	Displacement computation framework	62
4.11	Top view for the capsule motion inside the small intestine	63
4.12	Frame sequence where the PillCam SB3 capsule is going backwards	64
4.13	Backward displacement results	64
4.14	Frame sequence where the PillCam SB3 capsule is going forward	65
4.15	Forward displacement results	65
4.16	Per frame displacement (in percentage) in two examples of small bowel PillCam SB3 endoscopic capsule videos	66
4.17	Comparison between our sub-sampled results and the results obtained in <i>Given Imaging</i> software	67
4.18	Example of three frames generated snippet	68
4.19	Total loss	69
4.20	Depth results	70
4.21	3D pose estimation	70

List of Tables

2.1	Feature detectors (From Salahat and Qasaimeh (2017))	12
2.2	Feature detection algorithms (From Salahat and Qasaimeh (2017))	12
2.3	Advantages and disadvantages of Deep learning in Computer Vision	25
2.4	MobileNet architecture (From Howard et al. (2017))	33
3.1	PillCam SB2 and PillCam SB3 characteristics	47
4.1	HomographyNet Regression architecture	52
4.2	MobileNet-based HomographyNet architecture	53

Abbreviations and Symbols

VOVEC	Visual Odometer on Videos of Endoscopic Capsules
VEC	Video of Endoscopic Capsules
CE	Capsule Endoscopy
GI	Gastrointestinal
RF	radio-frequency
CHP	Centro Hospitalar do Porto
MOPS	Multi-Scale Oriented Patches
PCA	Principal Component Analysis
f	Focal Length
F	Fundamental Matrix
E	Essential Matrix
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
MLP	Multilayer Perceptron
FC	Fully connected
PET	Positron Emission Tomography
MRI	Magnetic Resonance Imaging
SLAM	Simultaneous Localization and Mapping
VSLAM	Visual Simultaneous Localization and Mapping
PTAM	Parallel Tracking and Mapping
SIFT	Scale-invariant Feature Transform
ORB	Oriented FAST and Rotated BRIEF
FAST	Features from Accelerated Segment Test
SURF	Speeded Up Robust Features
MSER	Maximally Stable Extremal Regions
RANSAC	Random Sample Consensus
DoF	Degrees of Freedom
DLT	Direct Linear Transform
FLANN	Fast Library for Approximate Nearest Neighbors
PSGG	Parameterized Sampling Grid Generator
DS	Differential Sampling
AOV	Angle Of View
fps	frames per second
AFR	Adaptative Frame Rate
DoF	Degrees of Freedom
MAE	Mean Absolute Error
SD	Standard deviation
RNN	Recurrent Neural Network
LSTM	Long Short-term Memory

Chapter 1

Introduction

An endoscopy is a medical procedure in which doctors use instruments to view the internal organs and vessels of the human body, allowing them to detect problems in a less invasive way. In 2001 the endoscopic capsule was introduced, creating a new innovative type of endoscopy. This swallowable and vitamin-size capsule takes advantage of a battery, a light and a tiny wireless camera to take thousands of pictures of the digestive tract in order to create a video. This camera is notably useful to see inside the small intestine - an area that is unreachable with more traditional endoscopic techniques, being more frequently utilized to perform follow-ups and when there is suspicion of ([Van de Bruaene et al. \(2015\)](#)):

- Obscure gastrointestinal bleeding;
- Crohn's disease;
- Hereditary polyposis syndrome;
- Small bowel tumor;
- Non-steroidal anti-inflammatory drug-induced small bowel lesions;
- Celiac disease.

Each exam produces a colour video with 8 to 10 hours of duration, which translates into a subjective, tedious and error-prone human-based diagnosis that takes typically between 60 and 90 minutes. Capsules do not provide localization information in the gastrointestinal track, even though their accurate localization is fundamental to identify abnormalities location, easing follow-up examinations and surgical interventions.

1.1 Motivation

Currently, physicians often estimate the location of the capsule visually, based on scarce landmarks, such as the duodenum and the ileocecal valve, and based on the apparent displacement that

can be perceived between video frames. Endoscopic systems can provide a more accurate capsule localization, but the patient is required to wear an additional piece of external hardware for the entire recording time. The most commonly used (commercially available) methods are based on radio-frequency (RF) sensor arrays. This external hardware entails several drawbacks ([Mateen et al. \(2017\)](#)):

- Patient discomfort;
- Cost increase;
- Bio-compatibility issues;
- Interference with the capsule camera sensor;
- Space limitations.

Software-only methods are also available and, despite some of them present potential to provide an accuracy comparable to that achieved by the methods exploiting external equipment, new and improved methods are still required.

This dissertation is proposed following up on the drawbacks that derive from the use of external hardware methods, which along with the recent improvements in most computer vision methods due to the disruptive field of deep learning, motivated us to solve some of the limitations of the current software-based methods.

1.2 Objectives

In this dissertation, we aim to produce software methodologies that improve the movement tracking of the endoscopic capsule inside the small-bowel, creating a visual odometry tool that retrieves position information from endoscopic images. This tool should be developed according to the actual needs of gastroenterologists and considering its practical application in clinical environment.

In order to do this, we first need to develop methods to detect and deal with non-informative frames, a crucial step when utilizing endoscopic capsules datasets.

Current software-based capsule localization methods generally rely on feature-based approaches, which perform poorly on endoscopic capsule videos that lack on distinct features. So, in order to improve upon this methods and overcome some of the challenges posed by endoscopic videos, we focus on methodologies from the disruptive field of deep-learning.

It is also important to note that our work starts by gathering a research summary of the existing endoscopic capsule localization techniques, documenting the most common difficulties researchers need to overcome when using a similar dataset.

1.3 Contributions

A summary of the contributions of this dissertation to the estimation of endoscopic capsules localization can be seen bellow. In this work we have:

1. Created a method that estimates the endoscopic capsule per frame displacement (in percentage) along the small bowel.
2. Implemented a method that estimates the endoscopic capsule 3D pose along the small bowel and that predicts a depth map for each frame of the endoscopic video.
3. Proposed two different methods to classify a endoscopic image as *informative* or *non-informative* for our localization algorithms.
4. Proposed a method to discard and deal with incorrect displacement predictions.
5. Successfully calibrated a PillCam SB3 endoscopic capsule camera.

1.4 Document Structure

This dissertation is composed by five chapters. Chapter 1 consist of an introduction where the context, motivation, objectives and contributions presented.

Chapter 2 presents the necessary theoretical concepts, mainly regarding multi-view geometry. Furthermore, classical computer vision techniques and deep learning methods are presented, as well as a brief review of the state of the art methods in endoscopic capsules position estimation.

In chapter 3 we start by characterizing our problem. Here, we also present a general system framework, then specifying the framework for each of both methods developed. Furthermore, we describe the dataset at our disposal and how we treat it to suit our end goals, as well as the camera calibration procedure.

In chapter 4 the implementation, results obtained and discussion for each of the developed methods are displayed.

In chapter 5 a final insight on this work is given. Here, we discuss the overall level of satisfaction according to the initially proposed goals and also present some recommendations and possibilities for future work.

Chapter 2

Literature Review

On this chapter we will mainly find information regarding multi-view geometry that we need to acknowledge in order to understand the solution presented for the problem in question. We will first talk about classic computer vision techniques (section 2.1.1), where we explore the basic math behind the simplified pinhole camera model (section 2.1.1.1), depth information recovery from images (section 2.1.1.2), feature detection, description and matching, as well as rectification (section 2.1.1.3) and finally, the homography (section 2.1.1.4). Then, we will approach deep learning (section 2.1.2), namely topics such as supervised (section 2.1.2.1) and unsupervised learning (section 2.1.2.2), convolutional neural networks (section 2.1.2.3) and deep homography estimation (section 2.1.2.4). Finally, a brief overview of the endoscopic capsule localization problem and methods will be presented (section 2.2).

2.1 Multiple View Geometry

Epipolar geometry is the stereo vision geometry, that is, describes the geometric relations when two images of different perspectives of the same scene are taken from two different locations (assuming the scene is rigid). This allows to extract 3D properties from a scene represented in a set of 2D images in a robust way, working around the intrinsic ambiguity problem (due to lost information) of the 2D to 3D mapping through a single image.

2.1.1 Classical Techniques

As seen in [de Oliveira \(2013\)](#), there are a few different ways to approach the subject of multiple view stereo imaging, such as depth-map merging, volumetric-based and feature point-based detection. Depth-map merging consists in finding correspondences between the depth-maps of two separate images ([Koch et al. \(2000\)](#); [Li et al. \(2010\)](#)). The volumetric-based methods firstly represent the scene as a set of 3D voxels and then proceeds to the energy minimization to decide if those voxels should be filled ([Vogiatzis et al. \(2007\)](#)). The feature point-based approach extracts and matches feature points, then reconstructing the 3D surface through geometric, photometric or visualization constraints ([Lhuillier and Quan \(2005\)](#); [Furukawa and Ponce \(2010\)](#)). Here, our

focus will be on the feature point detection methods (Liu et al. (2008); Ling et al. (2012); Rafiei and Saadatseresht (2013)), which can be generally described by the following sequence of steps, as seen in Kien (2005) and de Oliveira (2013):

1. Feature points search (section 2.1.1.3);
2. Robustly match the highest number of those feature points (section 2.1.1.3);
3. Use the result of the previous step to find the fundamental matrix F (equation 2.12) with the lowest inconsistency within the largest possible set of inliers (minimum error possible) (section 2.1.1.2);
4. Compute rectification homographies and rectify both views using the F matrix and the inlier matches (section 2.1.1.3);
5. Find the largest stereo matching possible with the highest confidence available for each match (section 2.1.1.3);
6. If found somewhere in the F matrix estimation or rectification algorithm, camera parameters should be used to return a quasi-Euclidean reconstruction. Else, the projective reconstruction should be returned;

This whole process is portrayed in Figure 2.1.

2.1.1.1 Camera Model

When light reflects on an object, it makes its way to our camera and is collected by our imager (image sensor). A simple, but useful camera model can be used to explain the geometry of this process, the pinhole camera model (Bradski and Kaehler (2008)). This is a commonly used model in the literature, as it simplifies the math when compared to a camera model that utilizes a lens in order to collect more light.

This model consists of a wall with a miniature hole at its center, so small that only allows a single ray to enter in each point of the scene, being the rest of them blocked. This point is then "projected" onto the projective plane (image plane), resulting in an always in focus image and an image size, relative to the object, given only by the camera focal length (Figure 2.2).

In Figure 2.2, we can see that by similar triangles:

$$-x = f \cdot \frac{X}{Z} \quad (2.1)$$

This can now be rearranged so that the math comes even easier and the image is not inverted on our projective plane (Figure 2.3).

Where now:

$$x = f \cdot \frac{X}{Z} \quad (2.2)$$

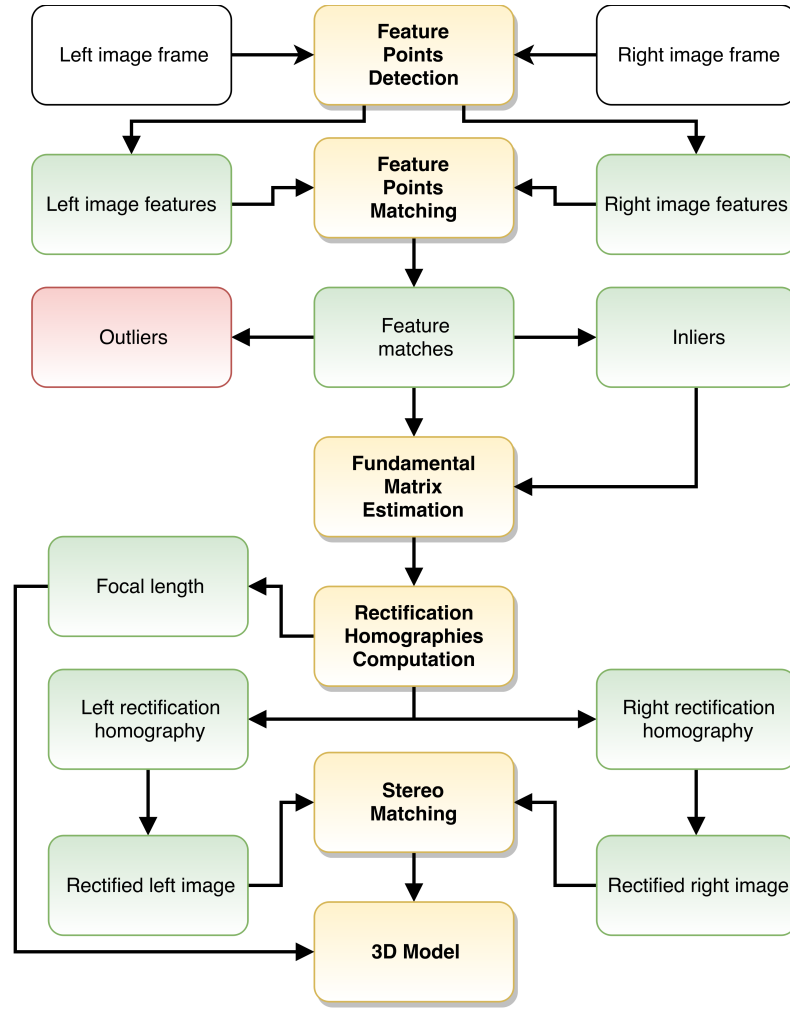


Figure 2.1: General progress for 3D reconstruction (as seen in [de Oliveira \(2013\)](#))

In this new, simplified model we assume that the point in the pinhole is the *center of projection*, that the point $q=(x, y, z)$ is the projection of the physical world point, $Q=(X, Y, Z)$, onto the image plane and that c_x and c_y are the coordinates for the optical center. This results in a pair of equations that represent the projection of a 3D point (X, Y, Z) onto the screen (image plane), at the pixel location of coordinates (x_{screen}, y_{screen}) :

$$x_{screen} = f_x \cdot \frac{X}{Z} + c_x \quad (2.3)$$

$$y_{screen} = f_y \cdot \frac{Y}{Z} + c_y \quad (2.4)$$

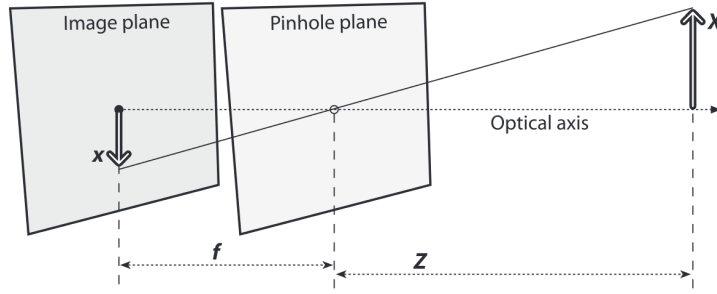


Figure 2.2: Pinhole camera model; X is a point in space and x its projection on the image plane, the Z axis is the optical axis and f the focal length (From Bradski and Kaehler (2008))

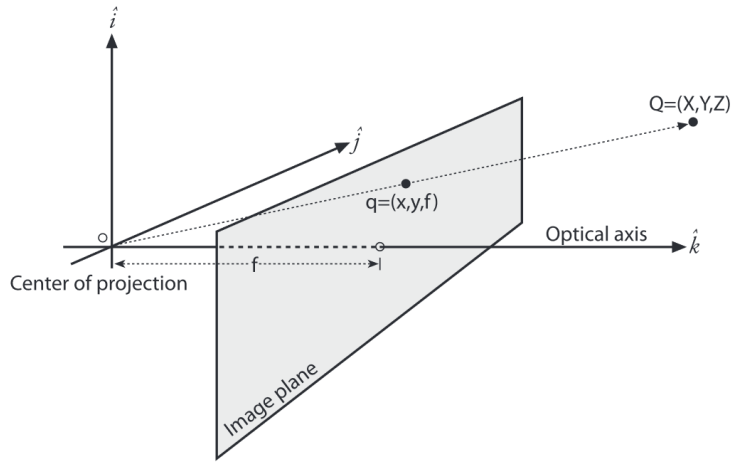


Figure 2.3: Rearranged pinhole camera model (From Bradski and Kaehler (2008))

If the image sensor is also not perfectly perpendicular to the optical axis, a skew (s) is created. Now, we can create our calibration matrix K (Szeliski (2010)):

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

We can consider a square sensor (making the focal length $f_x = f_y = f$) and $s = 0$ for various practical applications (Figure 2.4), so:

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

Having found the calibration or camera matrix parameters (camera intrinsics), we can bring it together with the camera orientation in space (camera extrinsics - characterized by a rotation

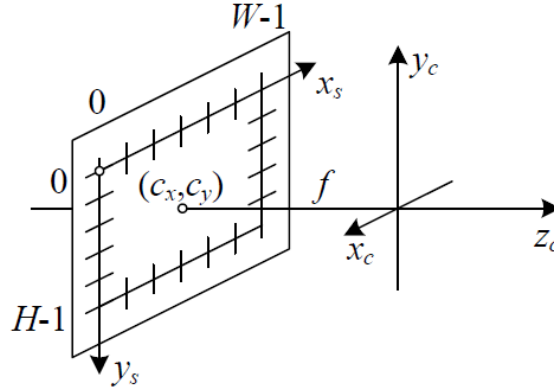


Figure 2.4: Simplified camera intrinsics - (c_x, c_y) is the optical center and W and H the image width and height (From [Szeliski \(2010\)](#))

matrix R and translation vector t) to create the camera matrix P :

$$P = K \cdot [R|t] \quad (2.7)$$

Which can also be presented as the following invertible matrix:

$$\tilde{P} = \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} \cdot \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \quad (2.8)$$

This new matrix allows us to map 3D coordinates, $\tilde{Q}=(X, Y, Z)$, to 2D image plane coordinates plus disparity $\tilde{q}=(x_{screen}, y_{screen}, 1, d)$, where:

$$\tilde{q} \sim \tilde{P} \cdot \tilde{Q} \quad (2.9)$$

being d the point's disparity and \sim indicating equality up to scale.

2.1.1.2 Structure from motion

As described in [Szeliski \(2010\)](#), *structure from motion* is the process to recover the 3D structure and pose from 2D image correspondences. In order to perform this estimate, our brain assesses the displacement objects suffer when seen through different angles. This is an analogous process to the one used in computer vision, where the disparity of similar matched features between frames is computed. The scene reconstruction is then possible if there is some knowledge between the cameras or if that knowledge is inferred (detailed in section 2.1.1.3).

Considering Figure 2.5, we can develop a mathematical model that describes the relation between views (as seen in [Szeliski \(2010\)](#) and [de Oliveira \(2013\)](#)). We will not specify how we got those relations, but the fully detailed mathematics can be examined in [Szeliski \(2010\)](#).

Having that $\hat{x}_j = K_j^{-1} \cdot x_j$ are the local ray directions (which cross the image plane at x or x_1) and knowing R and t , it is possible to arrive at a matrix that expresses the relation between

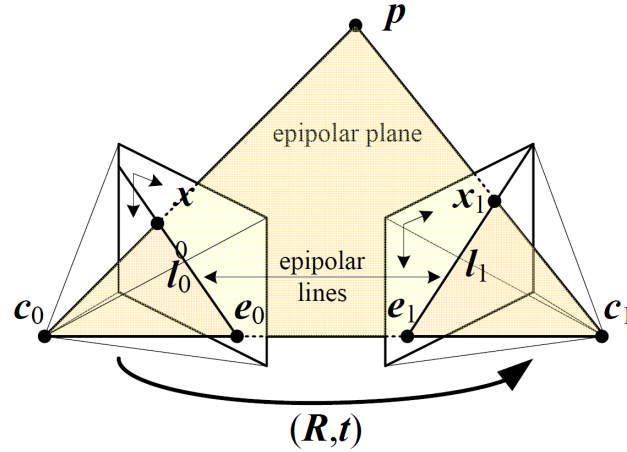


Figure 2.5: Two-view geometry based on the pinhole camera model (section 2.1.1.1). The camera centers are represented by c_0 and c_1 ; x and x_1 the projections of point p ; e_0 and e_1 the epipoles; l_0 and l_1 the epipolar lines, located at the epipolar plane; finally, the translation (t) and rotation (R) between both cameras are highlighted (From Szeliski (2010))

corresponding points in the image plane (equation 2.10). It is a 3×3 , rank 2 matrix named *essential matrix* and represented by E :

$$\hat{x}_1^T \cdot E \cdot \hat{x} = 0 \quad (2.10)$$

where

$$E = [t]_{\times} R \quad (2.11)$$

and with $[t]_{\times}$ being the skew-symmetric matrix.

However, this approach poses a problem if we rely on free-moving cameras for which we do not have any calibration information, as we do not know the values of R and t . The solution is to define the *fundamental matrix* F :

$$\hat{x}_1^T \cdot E \cdot \hat{x} = x_1^T \cdot K_r^{-T} \cdot E \cdot K_l^{-1} \cdot x = x_1^T \cdot F \cdot x = 0 \quad (2.12)$$

As described in de Oliveira (2013), the matrix defined above (in equation 2.12) is usually calculated with at least 8 point correspondences between two different image frames, employing *Singular Value Decomposition* (SVD) (Golub and Van Loan (2012)) to solve a system of equations. With more than 8 correspondences we can minimize the error of the estimation. It can also be calculated with only 7 matches, solving a non-linear system. Finally, there is the possibility of the existence of false-positive matches (outliers), which should be discarded, e.g using RANSAC (Fischler and Bolles (1981)) or even the method exposed in Olsen (1992) in case we have two rectified images (refer to section 2.1.1.3).

2.1.1.3 Local Features

As seen in Tuytelaars et al. (2008), local features (Figure 2.6) are image points or patterns, such as edges or even small image patches, that differ from its immediate neighborhood, being usually associated with a change of one, or more, image properties (e.g. intensity, color and texture).

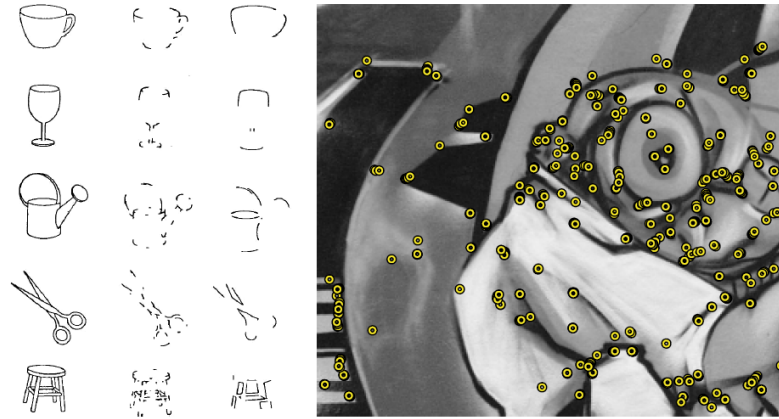


Figure 2.6: Features from a contour image, on the left, and a grayscale image, on the right, respectively (From Tuytelaars et al. (2008))

This interest points identification, together with feature descriptors, present, indeed, the potential to encapsulate the content of a frame. The feature descriptors convert the detected points into numerical descriptors, that provide a unique, condensed representation of these local features (Salahat and Qasaimeh (2017)). This process proved to be a powerful tool for a wide array of computer vision tasks, such as object detection, object tracking (Gauglitz et al. (2011)) and image features matching (Vincent and Laganier (2001)).

Feature Detection

The literature is filled with different detection and description algorithms, as well as surveys comparing their performance and different qualities (Tuytelaars et al. (2008); Liu et al. (2016); Lee and Park (2014); Mikolajczyk and Schmid (2005); Salahat and Qasaimeh (2017)). With that being said, there are still no ideal feature detectors, which means the decision of each one to use is based on the application needs. So, for instance, in an environment where it is impossible to know, *a priori*, the transformations that the scene might be exposed to, the ideal qualities for the features (and, consequently, feature detector algorithm) should be (de Oliveira (2013); Salahat and Qasaimeh (2017)):

- Distinctiveness: features should be distinctive enough, providing a useful amount of information;
- Locality: reduces the probability of features being blocked and allows simple geometrical model approximations (as seen in section 2.1.1.1);

- Accuracy: the features localization should be accurate through different scale and shapes;
- Quantity: the number of features detected should be large enough to represent the image content;
- Repeatability: the majority of the features detected in one view should be detected in another;
- Invariance: the algorithm should precisely model image deformations (e.g. rotation and scale) in order to minimize their effect on its detection task;
- Robustness: the sensitivity to small deformations, e.g. noise and blur, should be as small as possible.

Tables 2.1 and 2.2 show summarized information about state-of-the-art feature detectors and feature detection algorithms.

Category	Classification	Methods and Algorithms
Edge-based	Differentiation based	Sobel, Canny
Corner-based	Gradient based	Harris (and its derivatives), KLT, Shi-Tomasi, LOCOCO, S-LOCOCO
Corner-based	Template based	FAST, AGAST, BRIEF, SUSAN, FAST-ER
Corner-based	Contour based	ANDD, DoG-curve, ACJ, Hyperbola fitting, etc.
Corner-based	Learning based	NMX, BEL, Pb, MS-Pb, gPb, SCG, SE, tPb, DSC, Sketch Tokens, etc.
Blob (interest point)	PDE based	SIFT (and its derivatives), SURF (and its derivatives), CenSurE, LoG, DoG, DoH, Hessian (and its derivatives), RLOG, MO-GP, DART, KAZE, A-KAZE, WADE, etc.
Blob (key point)	Template based	ORB, BRISK, FREAK
Blob (interest region)	Segmentation based	MSER (and its derivatives), IBR, Salient Regions, EBR, Beta-Stable, MFD, FLOG, BPLR

Table 2.1: Feature detectors (From [Salahat and Qasaimeh \(2017\)](#))

Features Detector	Invariance			Qualities			
	Rotation	Scale	Affine	Repeatability	Localization	Robustness	Efficiency
Harris	■	-	-	+++	+++	+++	++
Hessian	■	-	-	++	++	++	+
SUSAN	■	-	-	++	++	++	+++
Harris-Laplace	■	■	-	+++	+++	++	+
Hessian-Laplace	■	■	-	+++	+++	+++	+
DoG	■	■	-	++	++	++	++
Salient Regions	■	■	■	+	+	++	+
SURF	■	■	-	++	+++	++	+++
SIFT	■	■	-	++	+++	+++	++
MSER	■	■	■	+++	+++	++	+++

Table 2.2: Feature detection algorithms (From [Salahat and Qasaimeh \(2017\)](#))

1. Edge-based Detectors

Edges take place at borders between regions with different color, intensity or texture. They tend to vary widely between different image views or even lighting condition changes, making them possibly undetectable after a rotation and/or a transformation, turning them not ideal for feature matching ([Szeliski \(2010\)](#)).

2. Corner-based Detectors

The most common definition of a corner in computer vision is given by [Harris and Stephens \(1988\)](#), being the *Harris corner detector* one of the most canonical algorithm for corner detection. This detector relies on the second-order derivatives matrix of the image pixel intensities, defined in two dimensions as the Hessian matrix around a point ([Bradski and Kaehler \(2008\)](#)):

$$H(p) = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \quad (2.13)$$

If a big change in the derivatives signal is found in both directions in a certain location, there is a high probability of it being a corner. So, basically, a corner is where we have two edges in at least two different directions centered around a point.

This algorithm was then modernized to be invariant to scale changes and, later, invariant to affine transformations ([Tuytelaars et al. \(2008\)](#)).

Another well-established algorithm is SUSAN (*Smallest Univalued Segment Assimilating Nucleus*), detailed in [Smith and Brady \(1997\)](#). This detector concept is simple: A circular mask with a fixed radius is defined around every pixel of the image, being the center pixel considered the nucleus; then, the intensity of the pixels inside the mask are compared to the intensity of the nucleus; those with brightness values identical to the nucleus are grouped, resulting in an area referred as USAN (*Univalued Segment Assimilating Nucleus*); finally, a corner is found at points where the amount of pixels in the USAN area drops below a defined threshold value.

Based on the algorithm mentioned above, emerged FAST (*Features from Accelerated Segment Test*) ([Rosten and Drummond \(2005\)](#)), which brings much more computational efficiency. While SUSAN needs to compute all the pixels inside a defined circle and compare them to the nucleus, FAST only needs to compare the pixels on the circumference (the highlighted squares in Figure 2.7). Denoting I_p by the center pixel intensity and T as a threshold value, the principle is: if the intensities of all the contiguous pixels in the mentioned circumference are superior to $I_p + T$ or inferior to $I_p - T$, then p is considered a corner ([Hassaballah et al. \(2016\)](#)).

As it is mentioned in [Rosten and Drummond \(2006\)](#), this high performance approach still has considerable limitations. These flaws were counteracted through machine learning (method detailed in [Hassaballah et al. \(2016\)](#)).

3. Blob Detectors

Three prominent blob-based algorithms are MSER (*Maximally Stable Extremal Regions*), SIFT (*Scale Invariant Feature Transform*) and SURF (*Speeded Up Robust Features*). As we can see in the surveys [Tuytelaars et al. \(2008\)](#), [Miksik and Mikolajczyk \(2012\)](#) and

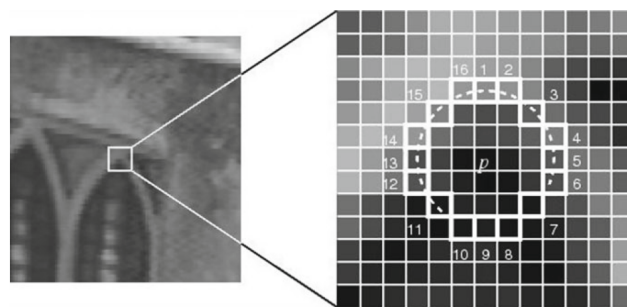


Figure 2.7: p is the center of a candidate corner; the highlighted squares represent the pixels used for the detection process; the dashed line represents the circumference of contiguous pixels; (From [Rosten and Drummond \(2006\)](#))

[Mikolajczyk and Schmid \(2004\)](#), the three of them show great promise regarding invariance and other qualities, as shown in Table 2.2.

SIFT ([Lowe \(2004\)](#)) is an *interest point-based blob feature detection algorithm*. It starts by collecting interest points by applying the *Difference-of-Gaussian* (DoG) and then selecting the local extrema (maxima and minima). These keypoints are represented as vectors that indicate scale, orientation and location (Figure 2.8).

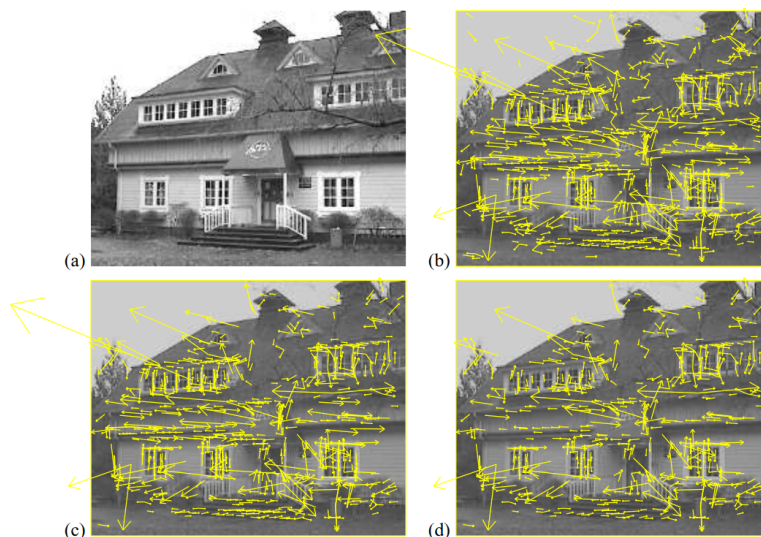


Figure 2.8: (a) original image; (b) interest points vectorial representations; (c) interest points remaining after the application of a contrast threshold; (d) final interest points after a threshold employment on the ratio of principal curvatures; (From [Lowe \(2004\)](#))

Due to its proliferation, a great deal of improvements on the SIFT algorithm were made, resulting in several derivative versions, such as PCA-SIFT ([Ke and Sukthankar \(2004\)](#)), n-SIFT ([Cheung and Hamarneh \(2007\)](#)), CSIFT ([Abdel-Hakim and Farag \(2006\)](#)) and SURF ([Bay et al. \(2006\)](#)).

SURF detector, when compared to SIFT, offers two big advantages, its speed and robustness. Its method consists in approximating Gaussian derivatives of second order (Hessian matrix) in a fast way, by means of integral images (or summed area table) using box filters. The approximated Hessian matrix determinant produces the blob response in a certain location in the image, which are stored in a blob response map. Finally, the keypoints considered are the local maxima that are detected and processed using quadratic interpolation. (Spyrou and Iakovidis (2012); Hassaballah et al. (2016); Tuytelaars et al. (2008))

MSER (Matas et al. (2004)) is a *region-based blob detector* that is invariant to affine transformations. In order to detect maximally stable extremal regions, binary regions are found by applying a threshold to the image in question at all the possible intensity levels - which implies the algorithm only works for grayscale images. The variations of the regions area size are measured between different intensity threshold values, being those with the minimal rate of change the regions considered as maximally stable. This is what confers them with affine geometric and photometric invariance.

As with the SIFT algorithm, MSER was also the base for different, extended and enhanced versions, e.g. MSER N-Dimensional Extension (Donoser and Bischof (2006); Vedaldi (2007); Forssén (2007)), Linear-Time MSER Algorithm (Nistér and Stewénus (2008); Alyammahi et al. (2015)), X-MSER (Salahat et al. (2015a); Salahat et al. (2017)) and The Parallel MSER Algorithm (Salahat et al. (2015b); Salahat et al. (2016)).

Feature Descriptors

The step after feature detection is feature matching, which determines the features that are in analogous locations in different views. This can be robustly done by defining a descriptor for each detected feature and then searching for descriptors correspondences through the different views captured. Typically, the image can suffer from some transformations in the form of translation and rotation, existing also the possibility of affine transformations occurrence. So, it is important that feature description algorithms are invariant to these changes. (Szeliski (2010); de Oliveira (2013))

Below we present some examples of feature descriptors found in the vast literature on the subject, as seen in Szeliski (2010):

MOPS (*Multi-Scale Oriented Patches*) (Brown et al. (2005)) is a descriptor that consists of simple bias and gain normalized 8×8 patches, sampled at a spacing of five pixels relative to the detection scale. It is this low sample frequency that allows the small sensitivity to feature point location error. The normalization turns the patch intensities mean to zero and their variance to one, making the features invariant to intensity affine changes.

In the SIFT descriptor algorithm (Lowe (2004)), features are obtained by calculating the gradient of each pixel in a 16×16 window around the detected interest point location, using the suitable level of Gaussian filter (on the left in Figure 2.9). Then, with a Gaussian fall-off function (blue circle in Figure 2.9), the gradient magnitudes are weighted, reducing the influence of those far from the center. To form the keypoint descriptor, gradient orientation histograms are created

over 4×4 sample regions of the previously calculated window, by softly adding the weighted gradients to one of eight histogram bins (intervals), being trilinear interpolation used. This results in a 4×4 array of orientation histograms with 8 orientation bins each, which are the constituents of the descriptor, a $4 \times 4 \times 8 = 128$ dimension vector (on the right in Figure 2.9).

Finally, this vector is normalized to unit length, its values capped to 0.2 and re-normalized to unit length.

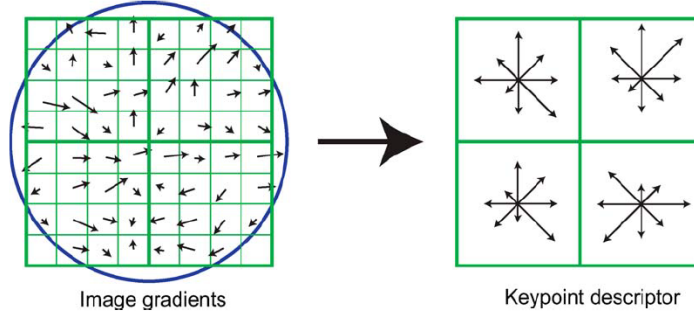


Figure 2.9: SIFT feature descriptor creation; This figure shows a 2×2 descriptor array calculated from a 8×8 pixel patch, although Lowe's experimental implementation uses a 16×16 pixel patch to calculate a 4×4 descriptor array (From [Lowe \(2004\)](#))

The PCA-SIFT descriptor ([Ke and Sukthankar \(2004\)](#)) - inspired by SIFT - presents a simplified way to compute the keypoint descriptors. It calculates the x and y gradients of a 39×39 pixel patch, resulting in a vector with a 3042 dimension. This vector has its dimensions reduced to 36 through principal component analysis (PCA, which is detailed in [Szeliski \(2010\)](#)).

SURF descriptor (*Speeded Up Robust Features*) represents the distribution of intensities around the keypoints. It creates a square area around those points and along their orientation, with the size of twenty times the scale at which each point detection occurred. Next, this area is divided into 4×4 smaller areas, which the Harr wavelet responses will be computed for, using integral images. Finally, to achieve rotation invariance, a dominant orientation is found, that occurs when the aggregated value of the Harr wavelet responses is maximum. ([Hassaballah et al. \(2016\)](#); [Spyrou and Iakovidis \(2012\)](#))

Rectification

Stereo systems configurations are rarely completely aligned, as both cameras hardly have fully co-planar and row-aligned image planes. That being said, it is when this condition is verified (as seen in Figure 2.10) that the computation of the stereo disparity is simpler, reducing also the stereo matching problem to a one-dimension search.

Taking Figure 2.10, it is possible to limit the search for point matches between the two images to the x-dimension, facilitating the disparity calculation. Considering a point in the real world that corresponds to one point on the left image plane, with horizontal coordinates x_l , and another on

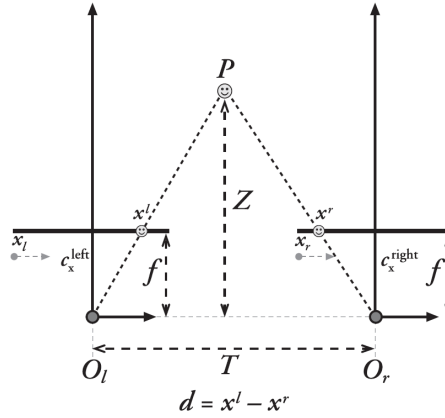


Figure 2.10: Disparity (d) calculation assuming a low baseline T ($T \ll Z$), assuming row-aligned image planes and that both cameras have the same focal length f . Z is the depth; T the displacement between cameras; the lines that go through the centers of projection (O_l and O_r) and the points of both image planes (c_l and c_r) are the principle rays of the imagers (From [Bradski and Kaehler \(2008\)](#))

the right image plane, with horizontal coordinates x_r , the disparity can be calculated between these two views ([Bradski and Kaehler \(2008\)](#)):

$$d = x_l - x_r \quad (2.14)$$

Also, by using similar triangles:

$$\frac{T - (x_l - x_r)}{Z - f} = \frac{T}{Z} \implies Z = \frac{f \cdot T}{x_l - x_r} \quad (2.15)$$

Now that we are aware of the importance of having the different views pre-aligned, we are going to introduce *rectification*, the process that makes it possible. In [Loop and Zhang \(1999\)](#), image rectification is described as the process of implementing homographies (explored further in section 2.1.1.4) to a couple of images whose epipolar geometry is known, such that the original image epipolar lines correspond to horizontally aligned lines in the transformed image. Loop and Zhang break down each of these homographies into three type of transforms: similarity, shearing and projective.

First, lets consider the homography H , given by:

$$H = H_a \cdot H_p \quad (2.16)$$

Being H_a a affine transform and H_p a projective transform. Then:

$$H_a = H_s \cdot H_r \quad (2.17)$$

where H_s is a shearing transformation and H_r a similarity transformation.

These transformations are done considering possible distortions, minimizing them through a *distortion minimization criterion*. This process, as well as its result, can be visualized in Figure 2.11. For a further detailed and complete description of this method please consult [Loop and Zhang \(1999\)](#).

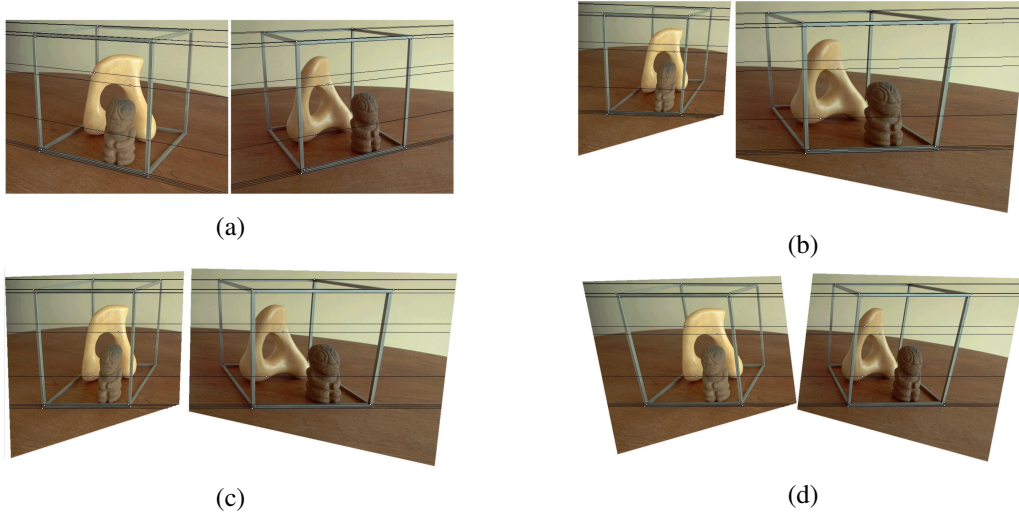


Figure 2.11: Results from the rectification algorithm presented by Loop and Zhang. **(a)** original images with a few epipolar lines superimposed; **(b)** The projective transformations (H_p and H'_p) result in the epipolar lines being parallel to each other in each image; **(c)** the similarity transformations (H_r and H'_r) resulting in horizontally aligned epipolar lines in both images; **(d)** the final image is the result of the shearing transformation (H_s and H'_s), which minimizes the horizontal distortion maintaining the images rectified; (From [Loop and Zhang \(1999\)](#))

In [Isgro and Trucco \(1999\)](#) a different approach is taken. Here, the algorithm proposed does not require the epipolar geometry and, more specifically, the fundamental matrix to be explicitly calculated. Instead, it takes advantage from the known form of the F matrix in a pair of rectified images (equation 2.18), in order to set up a cost function, minimizing it and obtaining the rectification homographies straight from image correspondences.

$$F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad (2.18)$$

Being H_1 and H_2 the homographies that rectify each image, we have that for each correspondent point p_1 and p_2 :

$$(H_2 \cdot p_2)^t \cdot F \cdot H_1 \cdot p_1 = 0 \quad (2.19)$$

In order to identify two matrices H_1 and H_2 that satisfy equation 2.19 and given N point correspondences (p_{1i}, p_{2i}) with $i = 1, \dots, N$, the following cost function is then minimized:

$$\mathcal{F}(H_1, H_2) = \sum_{i=1}^N [(H_2 \cdot p_{2i})^t \cdot F \cdot H_1 \cdot p_{1i}]^2 \quad (2.20)$$

As equation 2.19 is not a full rank matrix due to the null vector on the first row of F , there is still the need to find the first row of H_1 . This is achieved the minimization of the sum of square distances:

$$\sum_{i=1}^N [(H_1 \cdot p_{1i})_x - (H_2 \cdot p_{2i})_x]^2 \quad (2.21)$$

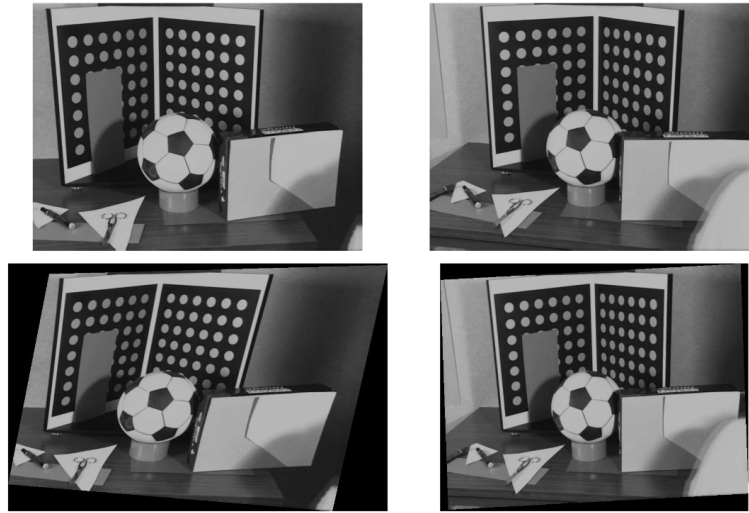


Figure 2.12: Results from the rectification algorithm presented by Isgrò and Trucco. We can see the original image pair on the top and the rectified image pair on the bottom (From [Isgrò and Trucco \(1999\)](#))

Feature Matching

Feature matching is a stereo matching process vital for a vast array of computer vision applications, such as image alignment, object recognition, motion tracking, 3D reconstruction, camera calibration and robot navigation ([Hassaballah et al. \(2016\)](#)). Its function is to establish correspondences between two image frames of the same scene or object. That being said, it is an error-prone procedure, mostly due to its ill-posed nature, which can be explained by the uncertainty caused by repetitive structures or patterns and by the match similarity measure randomness ([Šára \(2002\)](#)). This process, depicted in Figure 2.13, is initialized after the features are extracted, their descriptors computed and, optionally, the images rectified. It can also be expressed in a generalized way, as seen in [Hassaballah et al. \(2016\)](#):

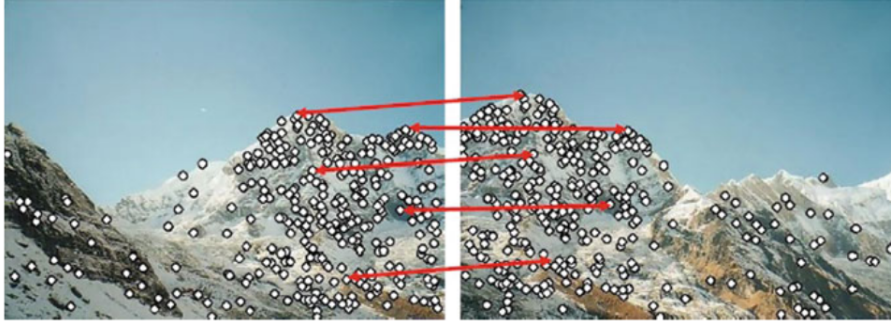


Figure 2.13: Feature matching between two images (From [Szeliski \(2010\)](#))

Considering p as point detected by a feature detector with an associated description $\phi_k(p)$ provided by a feature descriptor,

$$\Phi(p) = \{\phi_k(p) \mid k = 1, 2, \dots, K\} \quad (2.22)$$

which for all descriptors k ,

$$\phi_k(p) = (f_{1p}^k, f_{2p}^k, \dots, f_{n_k p}^k) \quad (2.23)$$

the goal is to define the most accurate correspondence q , a point in a different image from the group of N interest points detected $Q = \{q_1, q_2, \dots, q_N\}$, by correlating the description vector $\phi_k(p)$ with the description $\phi_k(q)$ of the points in Q . For this, the distance between the two descriptors is measured with equation 2.24,

$$d_k(p, q) = |\phi_k(p) - \phi_k(q)| \quad (2.24)$$

then, for each k descriptor, the points in Q are arranged in ascending order, resulting in the following set

$$\Psi_k(p, Q) = \{\psi_k(p, Q) \mid k = 1, 2, \dots, K\} \quad (2.25)$$

where

$$\psi_k(p, Q) = \{(\psi_k^1, \psi_k^2, \dots, \psi_k^N) \in Q \mid d_k(p, \psi_k^i) \leq d_k(p, \psi_k^j), \forall i > j\} \quad (2.26)$$

The *brute-force* algorithm is the most classical algorithm to solve the feature matching problem. It is fundamentally simple: it takes a descriptor from a feature in image one and compares it to each and every feature descriptor in image two, returning the closest one by determining their distance through some distance calculation function; this is repeated for all the pixels in image one.

It is clear that the *brute-force* method suffers from computation efficiency problems, as well as problems in similar textured images, where a unique optimal match is not possible ([de Oliveira](#)

(2013)). That being said, we can conclude that it is truly important to also determine efficient algorithms to match the features as swiftly as possible. Bearing this in mind and considering vector-based features, nearest-neighbor matching can be used. Adversely, the best nearest-neighbor algorithm and corresponding parameters rely upon the dataset attributes and the distance between the nearest-neighbor distance ratio must be below some defined threshold in order to prevent ambiguous matching candidates. For the job of matching high dimensional features in large datasets, FLANN (*Fast Library for Approximate Nearest Neighbors*) (Muja and Lowe (2009)) is one of the most efficient and commonly used implementation, providing a collection of multiple algorithms optimized for nearest-neighbor search. (Hassaballah et al. (2016))

Additionally, if the features are binary the previously described algorithms are not suited. For this type of features, the comparison is done through the Hamming distance, implementing a bit counter on the result of a XOR operation on the bits. In the case of the involvement of large datasets, an approximate matching algorithm can be used in order to speed up the process. (Hassaballah et al. (2016))

Finally, in order to remove outliers from matched features groups, statistically robust methods such as RANSAC (Fischler and Bolles (1981)) can be implemented during the fundamental matrix estimation as seen in Yang and Li (2013).

In conclusion, the performance of the different feature-based approaches that can be employed in order to solve the feature matching problem will greatly vary depending on some factors, such as the detected points properties and the chosen feature descriptor (Lindeberg (2015)). So, in order to maximize the potential of the matching algorithms it is of utter importance that the appropriate detectors and descriptors are used for each individual application. (Hassaballah et al. (2016))

2.1.1.4 Classical Homography Estimation

Homography

Homography has varied meanings throughout different sciences. The homography that concerns the field of Computer Vision, the *planar homography*, describes a projective mapping from an image plane to another (Bradski and Kaehler (2008)). Furthermore, the homography defines the relation between any two images of the same planar surface (assuming no distortion, using the pinhole camera model, detailed in section 2.1.1.1). That being said, it is present in numerous computer vision application, including image mosaicing (Brown et al. (2003)), virtual touring (Pan et al. (2004); Tang et al. (2007)), monocular SLAM (Shridhar and Neo (2015)) and 3D camera pose reconstruction (Zhang and Hanson (1996)). (Nguyen et al. (2017))

Usually, they are represented by a 3×3 matrix with 8 parameters:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad (2.27)$$

where the value of 1 is usually attributed to h_9 , due to the fact that the matrix has 9 elements, while the homography is represented only by 8.

Summing up, the mapping of a reference image point, p , into a corresponding target image point, p' , is given by the following expression:

$$p' = H \cdot p \quad (2.28)$$

considering that the two points $p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$ and $p' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$ are two dimensional and have homogeneous coordinates.

Homography Estimation Methods

There are two traditional approaches when it comes to homography estimation methods, direct and feature-based (Szeliski (2006)).

Direct methods, like the influential Lucas-Kanade algorithm (Lucas et al. (1981)), try to match each pixel by warping one image towards the other and comparing the pixel intensity through an error metric, e.g. the sum of squared differences. They begin with a guess for the homography parameters and then refine the values using a search or optimization techniques to minimize the cost function (e.g. gradient descent) (Baker and Matthews (2004)). The main obstacles with direct methodologies are the limited range of convergence and computational low efficiency due to the search procedure.

Following the direct methodology disadvantages, the most commonly used methods are, in fact, the feature-based. In these methods, the feature points are detected and described using a locally invariant feature detector and descriptor, such as SIFT (see section 2.1.1.3) or ORB (Rublee et al. (2011)), being the first one the benchmark algorithm and the second one a computationally faster approach. Then, the feature points that were detected are used to find at least 4 feature points matches across the image pair, being commonly used methods *Fast Library for Approximate Nearest Neighbors* (FLANN) based search and brute-force search (as seen in section 2.1.1.3). *Random Sample Consensus* (RANSAC) (section 2.1.1.3) is usually used to exclude the outliers and improve the matching process accuracy. As each match provides 2 degrees of freedom (DoF), we can calculate the homography 8 DoF given 4 matches using the *Direct Linear Transform* (DLT) (Hartley and Zisserman (2003)), which will now be presented as seen in Dubrofsky (2009):

Through the expansion of the equation 2.28, we have

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1u + h_2v + h_3 \\ h_4u + h_5v + h_6 \\ h_7u + h_8v + h_9 \end{bmatrix} \quad (2.29)$$

we then divide the first and second row by the third row, to obtain

$$-h_1u - h_2v - h_3 + u'(h_7u + h_8v + h_9) = 0 \quad (2.30)$$

$$-h_4u - h_5v - h_6 + v'(h_7u + h_8v + h_9) = 0 \quad (2.31)$$

Equations 2.30 and 2.31 can also be written as a matrix

$$A_i \cdot h = 0 \quad (2.32)$$

where

$$A_i = \begin{bmatrix} -u & -v & -1 & 0 & 0 & 0 & u'u & u'v & u' \\ 0 & 0 & 0 & -u & -v & -1 & v'u & v'v & v' \end{bmatrix} \quad (2.33)$$

and

$$h = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & 1 \end{bmatrix}^T \quad (2.34)$$

As stated earlier, each point correspondence produces two equations, so only four of this correspondences are necessary in order to find all eight parameters of the homography. Thus, given four point matches, we can stack their matrices A_1, A_2, A_3 and A_4 to get a 8×9 matrix A . Finally, taking into account that no 3 points can be collinear, the result of the homography can be found by solving the following equation:

$$A \cdot h = 0 \quad (2.35)$$

If the points correspondences used are exact, we will always get a matrix of rank 8, even if more than four correspondences are used. That being said, in practice, the points will not be exact and there will always be some uncertainty associated with the process, which can be minimized with a fitting cost function. Several cost functions can be used for this purpose, such as *Algebraic Distance*, *Geometric Distance*, *Reprojection Error* and *Sampson Error*, each described in Dubrofsky (2009).

To conclude, although feature-based methods show a big advantage when it comes to performance, they can also be inaccurate when they fail to find a satisfactory number of keypoints as a result of poor lighting conditions, substantial viewpoint differences between frames or images with a lack of distinct features. A full, generic feature-based algorithm is depicted in Figure 2.14.

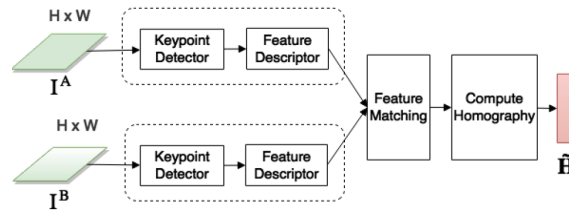


Figure 2.14: Generic feature-based algorithm (From Nguyen et al. (2017))

4-Point Homography Parameterization

Previously, the classical and simplest way to parameterize the homography was described, with a 3×3 matrix and a fixed scale (equation 2.27).

In Baker et al. (2006), new ways of parameterizing the homography are investigated in pursue of the maximization of plane estimation performance. One of the proposed solutions is the *4-point homography parameterization*, which according to the authors shows not only more robustness, as well as improved accuracy. This different parameterization method is based on one single type of location variable, the corner location, instead of separating the rotational and translational aspect of the transformation, as in the traditional approach (DeTone et al. (2016)).

The concept of this parameterization method is simple. Taking equation 2.28, each corner pixel $(u, v, 1)$ can be transformed by the 3×3 homography to calculate $(u', v', 1)$. Concurrently, if at least four points are available to calculate $\Delta u = u' - u$ and $\Delta v = v' - v$, it is possible to rebuild the 3×3 homography matrix used (Figure 2.15). This reconstruction can be achieved by utilizing the DLT algorithm described above.

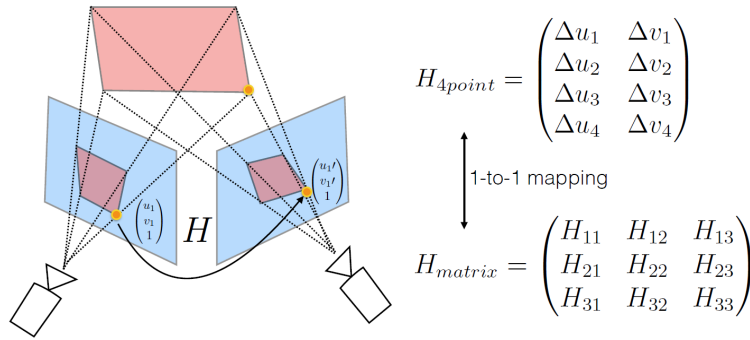


Figure 2.15: One-to-one mapping of the 4-point homography matrix to the 3x3 homography matrix (From DeTone et al. (2016))

2.1.2 Deep Learning Approach

As described in [LeCun et al. \(2015\)](#), the impact of machine-learning technology in our modern society has been growing rapidly, marking its presence on an increasing number of consumer products, such as smart-phones. It has been adopted in a broad number of application: web searches, advertising, content filtering on social networks, object detection and classification in images, speech recognition, among many others.

In recent years, the applications mentioned above have been shifting to a different class of techniques, labeled deep learning, which is becoming the state-of-the-art approach to most of this problems. This is due to the fact that conventional machine-learning systems are not capable of taking natural raw data (not previously processed and prepared) as an input.

Deep learning exploits representation-learning methods with multi-level representation. This methods allow the automatic detection of representations with raw data as the input starting point. Succinctly, it brings the representation from a low level (starts with raw input data) into a more abstract, higher level representation. This is achieved through the composition of simple, though non-linear modules, that if in enough quantity, open the possibility to learn very complicated functions and perform very intricate tasks. ([LeCun et al. \(2015\)](#))

Lastly, a list of general advantages and disadvantages of this new method, comparing to traditional computer vision methods, is presented in Table 2.3:

Advantages	Disadvantages
<ul style="list-style-type: none"> • State of the art method in many Computer Vision tasks • Automatic feature extraction, eliminating the need for <i>hand-crafted</i> features and reducing the heuristics on this problem • Flexible architectures allow an easy adaptation for different applications • Most successful method for solving large-scale problems 	<ul style="list-style-type: none"> • Usually requires a considerable amount of training data • Generally requires substantial computing power to train • Neural networks are challenging to analyze and debug, still being too much of a <i>black box</i> • Training difficulties (local minima problem, tuning hyper-parameters)

Table 2.3: Advantages and disadvantages of Deep learning in Computer Vision

Although we still need, generally speaking, a big amount of computational power to train big and deep neural networks, the tendency is for this to become less of an issue in the long run, as the computational power is an ever-growing resource that is becoming more accessible at the same time.

In the following sections we will briefly discuss the two different formats of machine and deep learning, then explore Convolutional Neural Networks (CNNs) and, finally, present the existing deep homography estimation methods.

2.1.2.1 Supervised Learning

The most frequent approach to machine and deep learning is, definitely, *supervised learning*. This approach can be divided into two main categories of problems: *classification* and *regression*. In classification problems a class label is the output, whereas in regression problems the output is a real number.

Now, assume that we have a set of images, containing, e.g. houses, dogs and persons. We want to classify each image, according to what they depict, putting them in one of the three categories mentioned above. As seen in [LeCun et al. \(2015\)](#), the following sequence shows the necessary steps to achieve the desired end result:

1. Collect a large and labeled dataset of images of dogs, houses and persons;
2. Feed the network with images and it outputs scores, one for each category;
3. We calculate a cost function that measures the error between the output scores and the desired scores (labeled data);
4. According to the mentioned above function results, the network internal parameters (weights) are modified in order to decrease the error;
5. Repeat steps 2 to 4 according to the number of epochs of training specified;
6. Feed the image to classify to the network and it will predict its category;

The process described from point 2 to 5 is called *training*. The fundamental idea behind training a network, while considering a dataset that contains labeled training examples, is rather simple. Basically, the objective is to minimize a cost function and to update the network internal parameters (weights) through the backpropagation algorithm (Figure 2.16), so that we get the actual output closer to the target output. These weights are used to calculate the representation in each present layer, based in the representation in the previous layer to that, that is, they define the input-output function. This is the process that allows networks to discover complex structures in some given data.

In order to make this weight vector adjustment, it is computed, for each weight, a gradient vector, which expresses by how much the error would increase or decrease if the weights were increased by a very small amount. The weight update is then carried through the opposite direction of the gradient vector, as the negative gradient indicates the steepest descent, i.e. taking the cost function closer to a minimum where the average error is lower. ([LeCun et al. \(2015\)](#))

Admitting that it is not the only procedure to accomplish this task, the most common among practitioners is the *stochastic gradient descent* (SGD). This algorithm can be briefly described in a few steps:

1. For a few examples of the input vector, the outputs and the error associated with them are computed, also calculating the average gradient for those randomly picked examples;

2. Adjust the weights appropriately;
3. Repeat this process for many small set of examples until the average of the cost function stops decreasing;

This is a rather simple method that quickly and consistently finds a good collection of weights when compared to other, more complex, optimization techniques. For a further detailed explanation please refer to [Goodfellow et al. \(2016\)](#) or [Bottou \(2010\)](#).

Usually, after the training process, the system is evaluated with a test dataset, different than the one we used to train the network. This helps to assess the level of generalization of the system, that is, how accurate are the network predictions when faced with inputs that were not used in the training process.

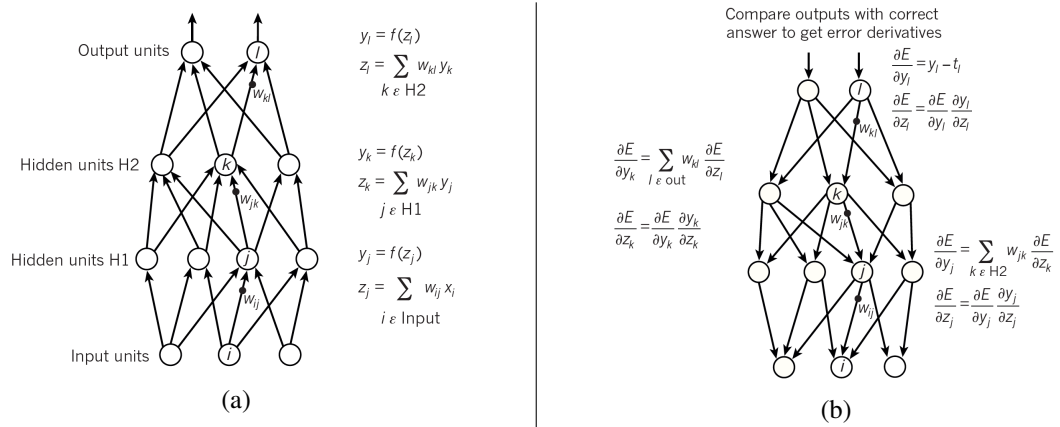


Figure 2.16: Forward and backpropagation in neural networks with multiple layers. **(a)** Forward pass computation in a neural network with two hidden layers (H1 and H2) and one output layer, each being a component where one can backpropagate gradients: at each layer level, the input (z) for each neuron is calculated, being given by a weighted sum of outputs of the neurons in the layer below (in this case, bias terms are omitted for the sake of simplicity); then, to compute the output, a non-linear function f is applied to z . the most common non-linear function are the *rectified linear unit* (ReLU) $f(z) = \max(0, z)$, the *hyperbolic tangent* $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ and the *logistic function* $f(z) = \frac{1}{1 + e^{-z}}$. **(b)** Backward pass computation: at each hidden layer level, we calculate the error derivative concerning the output of each unit through a weighted sum of the error derivatives respecting the total input of the neurons in the upper layer; then, the error derivative concerning the input is obtained by multiplying the error derivative concerning the output by the gradient of $f(z)$; finally, at the level of the output layer, the error derivative respecting the output neuron is obtained by calculating the gradient of the cost function; assuming this cost function is $\frac{(y_l - t_l)^2}{2}$, by its differentiation we get $y_l - t_l$, being t_l the target value. (From [LeCun et al. \(2015\)](#))

2.1.2.2 Unsupervised Learning

As we saw above, in supervised learning, the machine is given not only a set of inputs, but also a sequence of desired outputs (the data is labeled). Then, the objective is to learn to predict results accurately, given a new input, outputting either a class label or a real number.

Unlike in supervised learning, in unsupervised learning the focus is not to find an accurate prediction. Instead, the objective is to find compact data descriptions. In this different method, the machine simply receives inputs, not having access to supervised target outputs. Two classical examples of this unsupervised way of learning are clustering and dimensionality reduction. (Ghahramani (2004))

2.1.2.3 Convolutional Neural Networks

Since 2012, when Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton won the *ImageNet Challenge* (Russakovsky et al. (2015)) for a considerable margin by training a convolutional neural network (otherwise known as CNN or ConvNet), it steadily became mainstream and was acknowledged as the state of the art for many computer vision applications, such as image classification (Krizhevsky et al. (2012)) and semantic segmentation (Long et al. (2015)).

In LeCun et al. (1998), the authors presented not only a CNN for handwriting recognition (LeNet-5), but also exposed the problems related to the use of fully connected networks, also known as *multilayer perceptrons* (MLPs) for image classification. Unlike CNNs, MLPs have weights associated with every input, which becomes unmanageable for images with usable resolution, due to complexity and overfitting. Another problem is that they do not consider spatial distance between the input pixels, that is, it treats in the same ways pixels that are close and far from each other, making local features detection unfeasible. Concurrently, CNNs are architected to handle data that comes in the form of multiple arrays, such as 2D images containing pixel intensities represented in each of the three color channels. This exploitation of natural signals properties is based on four pillar ideas: local connectivity, shared weights, subsampling (pooling) and the use of many layers (LeCun et al. (2015), LeCun et al. (1998)).

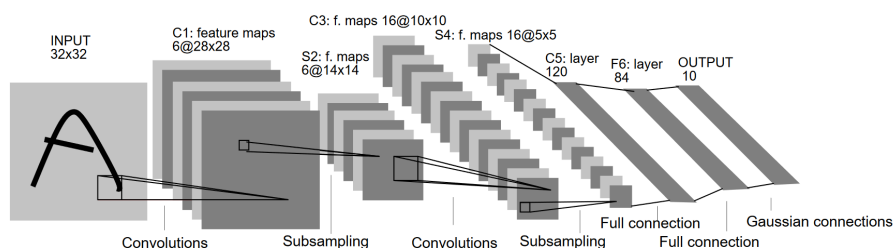


Figure 2.17: LeNet-5, an example of a convolutional neural network. Each plane represents a feature map (From LeCun et al. (1998))

As we can see in the example ConvNet shown in Figure 2.17, CNNs are based on four main ideas:

1. Convolutional layers;
2. Non linearity;
3. Subsampling or pooling layers;

4. Fully connected layers;

In each convolutional layer we have a set of filters that consist of trainable weight vectors (Figure 2.18). All of those filters are spatially small in height and width, but extend through the entire volume of the input layer (that is, in a $128 \times 128 \times 3$ color image we can use 3×3 filters, corresponding to $3 \times 3 \times 3 = 27$ weights for each neuron). During the forward-propagation process described in section 2.1.2.1 we convolve each filter (hence the network name) over the complete width and height of the input layer, being the output given by the dot products between the filter values and input layer values at all positions. This process will produce 2D feature maps, which provide the responses of each filter at every position. Regarding local connectivity, it is possible to say that each neuron will only be connected to a certain region of the input layer, being the spatial scope of this connection defined by the receptive field, which in turn is determined by the filter size. Finally, we can also add that the subsequent layers can combine previous layers detected features in order to detect higher level features.

Furthermore, the same detected features or patterns can appear in different regions of the input, being that the reason why filter weights are shared across different locations (weight sharing).



Figure 2.18: Example of filters learned by the first convolutional layer of ImageNet (From Krizhevsky et al. (2012))

The convolution result is then passed through a non-linearity. In the past, the standard non-linear activation function was the sigmoid, although the most common choice today is the simpler ReLU, as it does not saturate and also benefits from an increase of the learning rate (Krizhevsky et al. (2012)). For further detail refer to Figure 2.16.

Periodically and subsequently to the convolutional layers passed through a non-linearity, we have pooling layers performing the subsampling task. Concurrently to convolutional layers, that detect local features from the previous layer, pooling layers combine those features into one. Nowadays, the most popular type of pooling is max pooling (Figure 2.19), although some early works adopted mean pooling, e.g. in LeCun et al. (1998)). In the case of max pooling, we define a specific neighborhood and we take the largest value from each defined window, sliding that window through each feature map in order to create a new output map. There are three great advantages that come from this downsampling:

- It reduces the number of parameters and, consequently, lessens overfitting problems;
- Creates an invariance to small shifts and distortions;

- Creates an almost invariant to scale image representation. The fact that we can detect objects independently from their location is truly relevant, as an object can be located anywhere in the input image.

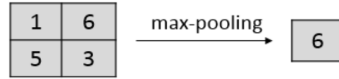


Figure 2.19: Max pooling operation example (From [Wu and Gu \(2015\)](#))

Finally, at the end of the network, we generally have fully connected (FC) layers. Their neurons are completely connected to the preceding layer activations and they output the desired predictions. In classification problems, they output the probability of belonging each class for the initial input. In regression problems they output N real numbers, being N the number of predictions we intend to make.

2.1.2.4 Deep Homography Estimation

As discussed in section 2.15, the homography estimation is vital in a multitude of computer vision tasks. We also saw that the traditional approaches for homography estimation are either slow or require a lot of complex feature matching, being under the risk of being inaccurate when certain conditions are not met. That being said, we will now present the most relevant work done to estimate the homography directly through deep learning methods, both supervised and unsupervised.

Supervised Learning Methods

When it comes to supervised learning methods, the most significant contribution comes from [DeTone et al. \(2016\)](#). Here, the authors propose two models of a convolutional neural network: a *Classification HomographyNet* and a *Regression HomographyNet* (Figure 2.20), both being architecturally similar to VGG networks ([Simonyan and Zisserman \(2014\)](#)).

As seen in Figure 2.20, these networks take two stacked gray-scale images as input (hence the $128 \times 128 \times 2$ input dimensions) that are related by a homography matrix. They work with 3×3 filters with Batch-Norm ([Ioffe and Szegedy \(2015\)](#)) and ReLUs (Figure 2.16) and are composed of 8 convolutional layers and two fully connected layers. After each two convolutions, a 2×2 max pooling takes place with a stride of 2. The first four convolutional layers include 64 filters each, whereas the last four include 128. A dropout is also applied with a probability of 50% before the first fully connected layer. Both networks remain comparable until the final layers. The regression network displays a 8×1 final layer (outputting the 4-point homography parameters discussed in section 2.1.1.4) and uses the Euclidean (L2) loss during training (Figure 2.20b):

$$\text{Euclidean (L2)} : \frac{1}{2} \cdot \|p(x) - q(x)\|^2 \quad (2.36)$$

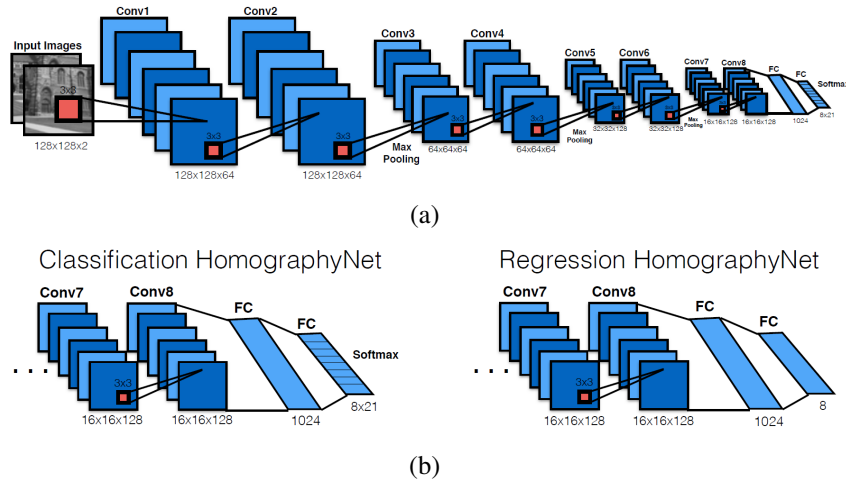


Figure 2.20: **(a)** *HomographyNet* - a deep convolutional neural network for homography estimation; **(b)** *Classification HomographyNet* head on the left and *Regression HomographyNet* head on the right (From DeTone et al. (2016))

The classification instance utilizes a quantization scheme, a Softmax activation function as the final layer and cross-entropy loss during training (Figure 2.20b):

$$\text{Crossentropy} : - \sum_x p(x) \cdot \log(q(x)) \quad (2.37)$$

being capable of generating a confidence score for each of the potential corners, due to the quantization error introduced by the use of 21 quantization bins for each of the output 8 dimensions (8×21 output). This is exemplified in Figure 2.21, where we can see how each 2D corner displacement can be represented as a distribution.

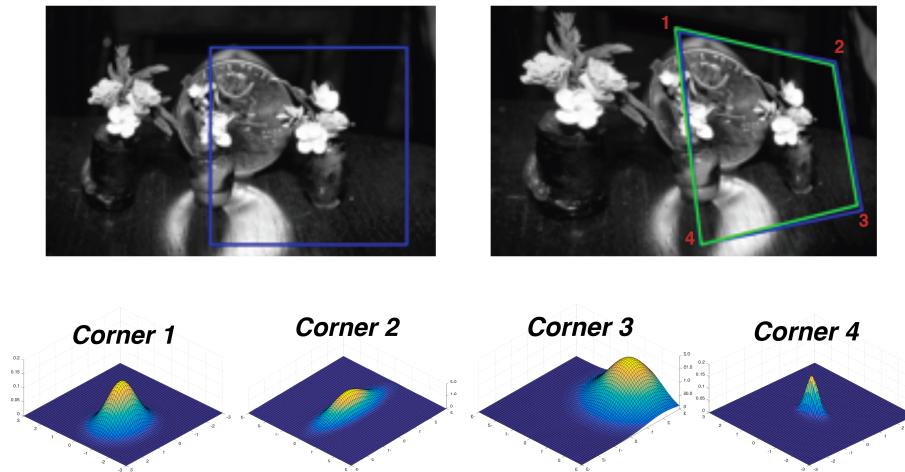


Figure 2.21: Corner Confidences Measure as a result of the *Classification HomographyNet* (From DeTone et al. (2016))

Inspired by DeTone et al. (2016), the github user Darwin Bautista proposed a MobileNet-based

architecture¹ (Howard et al. (2017)). MobileNets are light-weight models useful for a wide array of mobile and embedded vision applications, bearing an efficient architecture that leverages from depth-wise separable convolutions. They also dispose of two hyper-parameters that allow to size the model according to the problem demands.

A depth-wise separable convolution factorizes a standard convolution into a depth-wise convolution and a point-wise convolution. The first one applies only one filter to the each input channel and then the second one performs a 1×1 convolution to combine the first operation outputs.

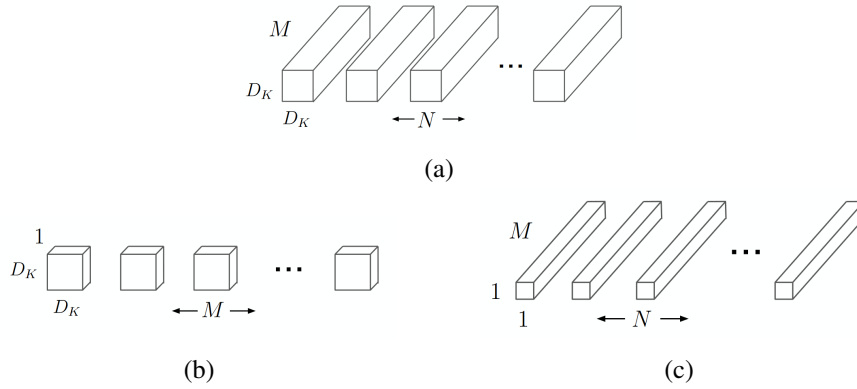


Figure 2.22: **(a)** Standard Convolutional Filters; **(b)** Depth-wise Convolutional Filters; **(c)** Point-wise Convolutional Filters (From Howard et al. (2017))

This technique brings great advantage in terms of computational cost:

A traditional convolutional layer outputs a feature map (assuming stride one and padding) that can be calculated as

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \quad (2.38)$$

which leads to a computational cost of

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \quad (2.39)$$

being F the $D_F \cdot D_F \cdot M$ input feature map, G the $D_F \cdot D_F \cdot N$ output feature map and K the convolutional kernel of $D_K \cdot D_K \cdot M \cdot N$.

On the other hand, a depth-wise convolution output feature map can be calculated as

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m} \quad (2.40)$$

which turns into a reduced computational cost of

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F \quad (2.41)$$

¹<https://github.com/baudm/HomographyNet>

The traditional MobileNet architecture is shown in Table 2.4. That being said, in the Homography estimation MobileNet-based architecture proposed by Darwin Bautista, the Fully Connected and Softmax layers were replaced by 4 convolutional layers, which are concatenated together and introduced to a Fully Connected layer (as seen in Figure 2.23)

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5×	Conv dw / s1	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 512$
	Conv dw / s2	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 1024$
	Conv dw / s2	$3 \times 3 \times 1024$ dw
	Conv / s1	$1 \times 1 \times 1024 \times 1024$
	Avg Pool / s1	Pool 7×7
	FC / s1	1024×1000
	Softmax / s1	Classifier

Table 2.4: MobileNet architecture (From Howard et al. (2017))

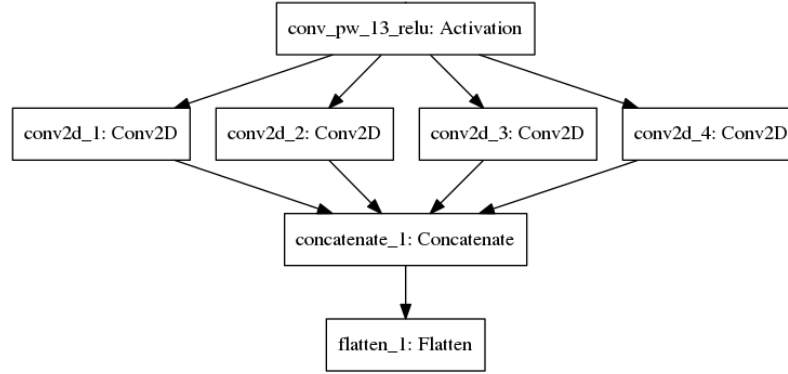


Figure 2.23: Homography estimation MobileNet-based architecture head

Unsupervised Learning Methods

To the best of our knowledge, the only work that treats the homography estimation problem as an unsupervised learning method is by Nguyen et al. (2017). Being unsupervised, it performs a pixel-wise intensity error metric minimization, that presents the great advantage of not requiring label data.

Inspired by direct methods (refer to section 2.1.1.4), [Nguyen et al. \(2017\)](#) devised a loss function, as follows:

$$L_{PW} = \frac{1}{|p_i|} \cdot \sum_{p_i} |I^A(\mathcal{H}(p_i)) - I^B(p_i)| \quad (2.42)$$

being I^A and I^B an image pair with pixels of homogeneous coordinates $p_i = (x_i, y_i, 1)^T$ and $\mathcal{H}(p_i)$ pixel coordinates warped by the 4-point homography (\tilde{H}_{4point}) seen in section 2.1.1.4. The goal is to output the \tilde{H}_{4point} that minimizes the L1 photometric loss described in equation 2.42.

Alike the supervised method described in section 2.1.2.4, a VGG-like architecture was chosen for the regression model. This is depicted in the first segment of Figure 2.24. That being said, the big contribution of this work comes from the differentiable layers in the second segment of Figure 2.24, that must remain differentiable in order to allow training through backpropagation. Also, as the error depends on pixel intensities differences, instead of differences on the \tilde{H}_{4point} values as in the supervised method in section 2.1.2.4, training is not as easy and stable. Furthermore, the use of a pixel-wise photometric loss presuppose an invariance of intensity and contrast values between images. In order to counteract this problem, [Nguyen et al. \(2017\)](#) standardize the images by the mean and variance of the pixel intensities and inject random illumination shifts across all the dataset.

The inputs for the unsupervised model consist in two 128×128 stacked patches (P^A and P^B) cropped from images I^A and I^B , four corners of image I^A represented by C_{4pt}^A and image I^A itself.

The second segment of Figure 2.24 starts with a *Tensor Direct Linear Transform* (Tensor DLT) that allows the calculation of the mapping from the 4-point homography representation (\tilde{H}_{4point}) to the 3×3 homography representation (\tilde{H}). A Tensor DLT is a regular DLT applied to a tensor so that it remains differentiable (refer to section 2.1.1.4 for a detailed DLT explanation). C_{4pt}^A and \tilde{C}_{4pt}^B are the inputs and \tilde{H} the output.

Then, a *Spatial Transformation Layer* (inspired by [Jaderberg et al. \(2015\)](#)) is introduced, applying \tilde{H} to the pixel coordinates (p_i) of image I^A in order to obtain warped coordinates ($\mathcal{H}(p_i)$), while still being differentiable. To achieve this, a three-stage inverse warping process is performed:

1. The normalized inverse \tilde{H}_{inv} is computed
2. *Parameterized Sampling Grid Generator* (PSGG)
3. *Differential Sampling* (DS)

In the first step, we start by normalizing the pixel coordinates of I^A and I^B , such that they stay in a range between $[-1, 1]$. Therefore, to obtain \tilde{H}_{inv} :

$$\tilde{H}_{inv} = M^{-1} \cdot \tilde{H}^{-1} \cdot M \quad (2.43)$$

being M :

$$M = \begin{bmatrix} \frac{W'}{2} & 0 & \frac{W'}{2} \\ 0 & \frac{H'}{2} & \frac{H'}{2} \\ 0 & 0 & 1 \end{bmatrix} \quad (2.44)$$

where W' and H' are I^B width and height, respectively.

In the second step, a grid of the same size as I^B is created ($G = \{G_i\}$), such that each element of this grid (G_i) is represented by a set of image I^B pixel coordinates. Then, the inverse homography (\tilde{H}_{inv}) is used to these coordinates a new grid ($\mathcal{H}_{inv}(G)$) with pixels of image I^A is obtained:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \tilde{H}_{inv} \cdot \begin{bmatrix} u'_i \\ v'_i \\ 1 \end{bmatrix} \quad (2.45)$$

Finally, in the last step, utilizing the points of the newly obtained grid, a sampled warped image V (H', W') with C channels is formed:

$$V_i^C = \sum_n \sum_m I_{nm}^C \cdot k(u_i - m; \Phi_u) \cdot k(v_i - n; \Phi_v), \forall i \in [1 \dots H'W'], \forall c \in [1 \dots C] \quad (2.46)$$

being W and H the width and height of image I^A , k the sampling kernel with Φ_u and Φ_v as the image interpolation parameters, I_{nm}^C the value of the pixel located at (n, m) in channel C of the input image. The output of this equation (V_i^C) is the value new image pixel located at (u_i, v_i) in channel C . As bilinear interpolation is used, equation 2.46 turns into:

$$V_i^C = \sum_n \sum_m I_{nm}^C \cdot \max(0, 1 - |u_i - m|) \cdot \max(0, 1 - |v_i - n|) \quad (2.47)$$

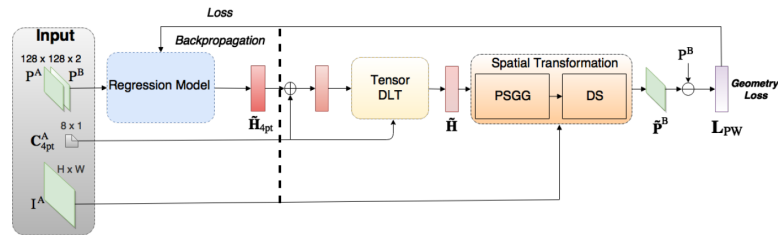


Figure 2.24: General view of the homography estimation unsupervised model (as seen in [Nguyen et al. \(2017\)](#))

2.1.2.5 Depth and Ego-Motion Estimation

The visual odometry problem can be tackled directly and in an end-to-end fashion by training a network that consists in the combination of a depth prediction network and a pose prediction network. This combo has been proposed several times and in different forms in the literature:

in supervised/unsupervised approaches and using monocular/stereo datasets. For example, [Umenhofer et al. \(2017\)](#) presented *DeMoN*, where a network is trained in a supervised way and with monocular videos in order to predict depth and motion from sequential image pairs. Another method that allies this two type of predictions is *UnDeepVO* by [Li et al. \(2017\)](#), where stereo videos are used to train a network that can predict depth and camera pose from monocular videos. Then, [Zhou et al. \(2017\)](#) proposed a method that can be trained with monocular videos in a completely unsupervised way in order to infer depth and camera pose. This last technique is going to be detailed in section 2.1.2.5.

Unsupervised Learning Method

The work by [Zhou et al. \(2017\)](#) titled SfMLearner, trains a model (seen in Figure 2.25) that observes image sequences to predict probable camera motion in the form of 6-DoF transformation matrices and scene structure in the form of per-pixel depth maps (Figure 2.26a). As said before, it is trained in a completely unsupervised manner and with monocular image sequences. Although the depth and pose networks need to be trained simultaneously, depth and pose can be later inferred separately (Figure 2.26b).

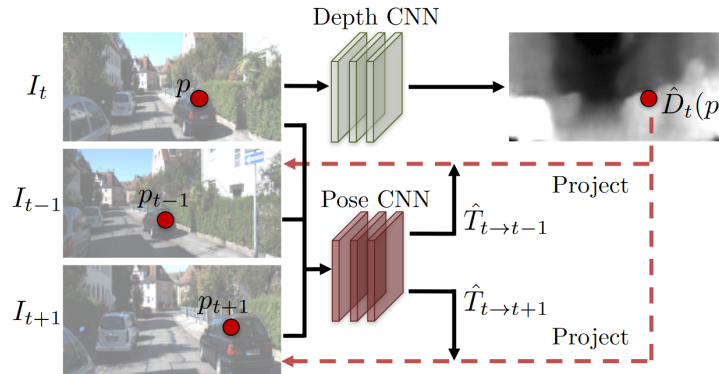


Figure 2.25: General SfMLearner training pipeline based on view synthesis. The DepthNet takes in the target-view RGB image (I_t) and outputs a per-pixel depth map (\hat{D}_t). The PoseNet takes not only I_t , but also a specified number of source views (e.g. I_{t+1} and I_{t-1}), outputting the corresponding camera transformation matrices (e.g. $\hat{T}_{t \rightarrow t+1}$, $\hat{T}_{t \rightarrow t-1}$). Finally, both networks outputs are used to inversely warp the source-views into the target-view in order, being the photometric reconstruction loss used to train the networks in a completely unsupervised way (From [Nguyen et al. \(2017\)](#))

The major supervision signal in [Zhou et al. \(2017\)](#) work stems from the task of *novel view synthesis* ([Flynn et al. \(2016\)](#); [Xie et al. \(2016\)](#)), that is, the task to create synthetic images of a scene from different camera poses given a reference view input. This method innovates by removing the necessity to obtain labels, proposing an approach to view-synthesis that does not require any pose groundtruths. That being said, in order to synthesize a target view we need a per-pixel depth map of the target image, alongside with camera pose and visibility in nearby views (e.g. previous and next frame). Finally, having an image sequence with N frames with one of

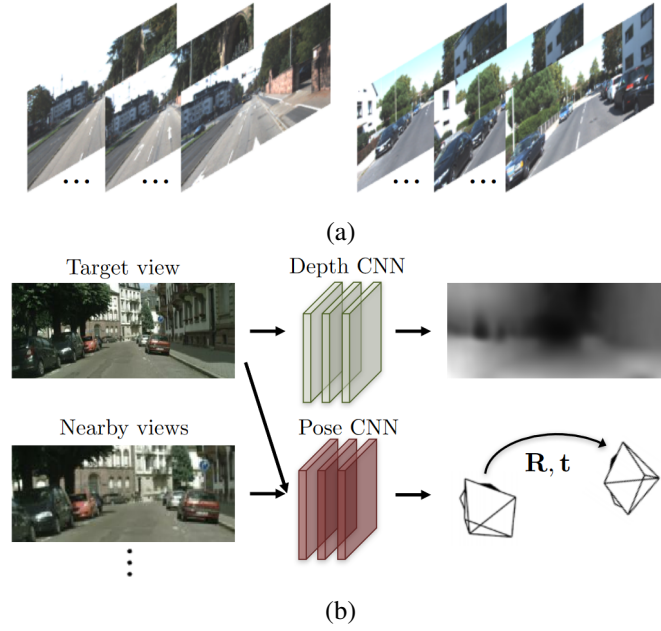


Figure 2.26: **(a)** The training data consists of video frames sequences; **(b)** Both depth and pose networks can be tested separately; (From Zhou et al. (2017))

them being the target view (I_t) and the rest of them source views (I_s), we can formulate the view synthesis as:

$$\mathcal{L}_{vs} = \sum_s \sum_p |I_t(p) - \hat{I}_s(p)| \quad (2.48)$$

where p denotes pixel coordinates, \hat{I}_s is the source-view (I_s) warped to the target-view (I_t) coordinate frame. This warping transformation requires the predicted target frame depth (\hat{D}_t), the predicted 4×4 camera transformation matrix ($\hat{T}_{t \rightarrow s}$) and the source-view frame. So, we can obtain the projection of a target-view pixel (p_t) onto the source view (p_s) as follows:

$$p_s \approx K \cdot \hat{T}_{t \rightarrow s} \cdot \hat{D}_t(p_t) \cdot K^{-1} \cdot p_t \quad (2.49)$$

being K the camera intrinsic matrix and \hat{D}_t and $\hat{T}_{t \rightarrow s}$ the predicted depth map and predicted relative camera pose, respectively. It is also extremely important to notice that the computed p_s coordinates are continuous values. So, in order to be able to compute the pixel values in a differentiable manner, Zhou et al. (2017) used the differentiable bilinear sampling mechanism presented in Jaderberg et al. (2015), as so:

$$\hat{I}_s(p_t) = I_s(p_s) = \sum_{i \in \{t,b\}, j \in \{l,r\}} w^{i,j} \cdot I_s(p_s^{i,j}) \quad (2.50)$$

with $w^{i,j}$ being linearly proportional to the distance between p_s and $p_s^{i,j}$ and with $\sum_{i,j} w^{i,j} = 1$. This warping mechanism is clearly illustrated in Figure 2.27.

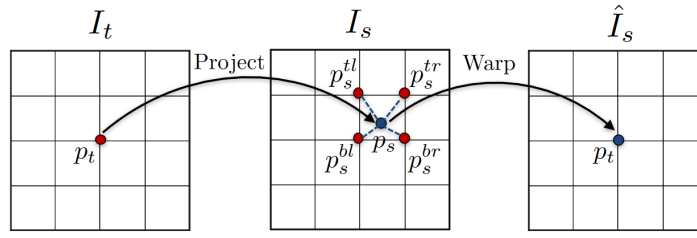


Figure 2.27: Differentiable warping process. First, p_t is projected onto the source-view based on the predicted depth map and camera pose; then, the bilinear interpolation is used to compute the value of the warped image \hat{I}_s at coordinates p_t (From [Nguyen et al. \(2017\)](#))

It is also important to notice that when this synthetic view is exercised on monocular videos or image sequences it is implicit that:

1. The scene is static;
2. There is no occlusion or disocclusion between the target and the source-views;
3. The captured surface is Lambertian (appears uniformly bright from all viewing directions), so that the photo-consistency is relevant;

With this limitation in mind, [Zhou et al. \(2017\)](#) proposed to improve the robustness of their training method by also training a *explainability prediction network* in simultaneous with the other two CNNs. This new network outputs a per-pixel soft mask (\hat{E}_s) for each target and source training pair in order to express the network belief in the success of the view synthesis prediction for each target pixel, resulting in the following view-synthesis objective:

$$\mathcal{L}_{vs} = \sum_{\langle I_1, \dots, I_N \rangle \in \mathcal{S}} \sum_p \hat{E}_s(p) |I_t(p) - \hat{I}_s(p)| \quad (2.51)$$

As there is no direct supervision for the explainability task, the predicted \hat{E}_s would always be zero, perfectly minimizing the loss in equation 2.51 and inhibiting training. So, in order solve this problem, a regularization term $\mathcal{L}_{reg}(\hat{E}_s)$ is added, stimulating non-zero predictions by minimizing the cross-entropy loss with a constant label of 1 for each pixel. Several illustrative examples can be seen in Figure 2.28.

The last problem found in the training method is that being the gradients mainly obtained by deriving pixel intensity differences between $I(p_t)$ and the four neighbours of $I(p_s)$, training would be very challenging if the correct p_s is found in low-texture area or even far from the current estimation. The proposed strategy to reduce this issue is to define a multi-scale and smoothness loss ([Garg et al. \(2016\)](#); [Godard et al. \(2017\)](#)), allowing gradients to be derived from larger spatial areas directly. So, for smoothness, the $L1$ norm of the second-order gradients of the predicted



Figure 2.28: Explainability masks examples where highlighted pixels are branded unexplainable. From rows 1 to 3 motion occurs and from rows 4 to 5 occlusion or disocclusion occurs (as seen in [Zhou et al. \(2017\)](#))

depth-maps was chosen to be minimized, encouraging smoothly changing depth values. Finally, all of this results into the following final loss function:

$$\mathcal{L}_{final} = \sum_l \mathcal{L}_{vs}^l + \lambda_s \cdot \mathcal{L}_{smooth}^l + \lambda_e \cdot \sum_s \mathcal{L}_{reg}(\hat{E}_s^l) \quad (2.52)$$

where l is the index of different image scales and s the index for source images, while λ_s and λ_e are the parameters to control the weight of the depth smoothness loss and the explainability regularization loss.

For the depth network the DepthNet architecture from [Mayer et al. \(2016\)](#) was adopted, establishing an encoder-decoder with skip connections and multi-scale side predictions, as shown in Figure 2.29a. This module is capable of outputting a per-pixel depth map given a single RGB image frame. Every convolutional layer is followed by the ReLU activation function with the exception of prediction layers, where the activation function is given by $1/\alpha \cdot \text{sigmoid}(x) + \beta$ with $\alpha = 10$ and $\beta = 0.01$ in order to obtain a predicted depth always positive and in an acceptable range.

Unlike DepthNet, PoseNet and the Explainability network (seen in Figure 2.29b) take as input a concatenation of the target-view and the source-views along the color channels. The pose-only part of the module consists of 2 convolutional layers with a stride of 7 followed by a convolutional layer with a 1×1 kernel that outputs $6 \cdot (N - 1)$ channels (3 translation values for each source view and 3 Euler angles for each source-view, being N the number of input frames for only one training sequence). In the end, a global average pooling is used to aggregate the prediction from all spatial

locations. The last convolutional layer has no nonlinear activation applied, being the rest followed by a ReLU activation function.

The explainability-only part of the prediction module depicted in Figure 2.29b is composed by 5 deconvolutional layers with multi-scale side predictions. The prediction layers have no non-linear activation applied, being applied to the rest of deconvolutional layers a ReLU activation function. It presents $2 \cdot (N - 1)$ output channels for each prediction layer, being every other channel normalized by a softmax function in order to compute the explainability prediction for each source and target-view pair.

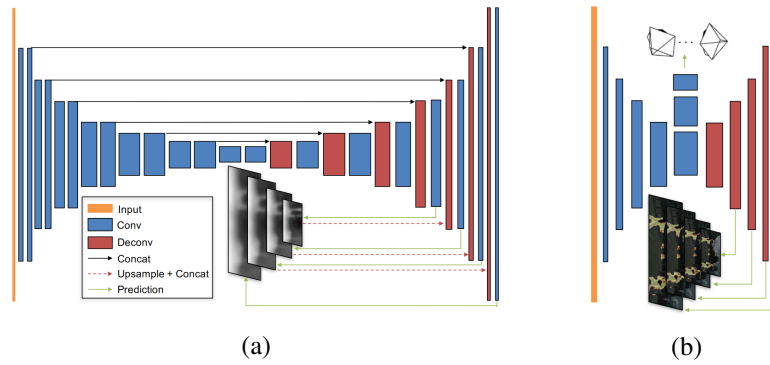


Figure 2.29: **(a)** DepthNet architecture. The width of each rectangular block indicates the number of output channels and the height indicates the dimension of each feature map in the respective layer. Also, a reduction or increase in the rectangular blocks indicate a change by the factor of 2. The kernel size for the first four convolutional layers is, in order, 7, 7, 5, 5, being for the rest of size 3. Finally, the number of output channels in the first convolutional layer is 32; **(b)** PoseNet and Explainability architecture. The first layers are commonly shared between both architectures, although they are then divided in two so that each of them can predict what is meant to predict. The kernel size for the first two convolutional layers and the last two deconvolutional and prediction layers is, in order, 7, 5, 5, 7, being for the rest of size 3; (From [Zhou et al. \(2017\)](#))

2.2 Endoscopic Capsules Position Estimation

Since its initial development by Given Imaging and its approval in Western countries in 2001, capsule endoscopy (CE) has suffered substantial improvements, still being in a continuous state of development ([Nakamura and Terano \(2008\)](#)). Different from traditional endoscopy it is non-invasive and pain-free, and thus is more convenient for long duration gastrointestinal screening ([Turan et al. \(2017b\)](#)). Although it is more commonly used for obscure gastrointestinal (GI) bleeding, it presents great potential for other diseases diagnosis or follow-up applications, such as Crohn's disease, small bowel tumor, hereditary polyposis syndromes and celiac disease; this is given to its ability to access difficult body parts (e.g. small intestines) when compared to standard endoscopy ([Van de Bruaene et al. \(2015\)](#)).

One crucial aspect in order to ensure the progression and wider adoption of this medical instrument is the development of endoscopic capsule localization methods, which can be explained by the following factors:

- The resulting video duration is usually between 8 and 10 hours. With only a few landmarks available in the entire gastrointestinal (GI) track, even obtaining a very general, organ-wise localization is challenging. This leads to a lengthy and error-prone manual reviewing process;
- If any abnormality is found during the GI track screening, a precise capsule localization is of vital importance, not only for the medical diagnosis but also for any necessary surgical interventions;
- A few different groups have suggested endoscopic capsules prototypes with remote control abilities and some additional features, such as biopsy and local drug delivery (Goenka et al. (2014); Munoz et al. (2014); Carpi et al. (2011); Keller et al. (2012); Mahoney et al. (2013); Yim et al. (2014); Petruska and Abbott (2013)). That being said, capsule motion control heavily relies on a precise real-time localization (Turan et al. (2017b)).

2.2.1 Capsule Localization Methods Overview

In the literature we can find several proposed methods that try to solve the endoscopic capsule localization problem. We will now present a summarized evolution of those methods as seen in Turan et al. (2017b), Sitti et al. (2015), Iakovidis and Koulaouzidis (2015) and Mateen et al. (2017), showing some of the most relevant work done in this field.

Ultrasonic imaging (Fluckiger and Nelson (2007); Rubin et al. (2007); Kim et al. (2008)), fluoroscopy (Carpi et al. (2011); Than et al. (2012)), magnetic resonance imaging (MRI) based techniques (Than et al. (2012); Krieger et al. (2011)), positron emission tomography (PET) based techniques (Than et al. (2012)), radio transmitter based techniques (Umay (2015); Dey et al. (2017)) and magnetic field based techniques (Yim and Sitti (2013); He et al. (2015); Hu et al. (2005); Pham and Aziz (2014)) are all methods that require extra hardware implementations beyond the basic camera sensor. Most commonly, commercially available methods are based on external radio-frequency sensor arrays that receive the signals broadcasted by the endoscopic capsule, with an average error of the 3D localization of 13.26 cm^3 ($2.00 \pm 1.64 \text{ cm}$ in x axis, $2.64 \pm 2.39 \text{ cm}$ in y axis and $2.51 \pm 1.83 \text{ cm}$ in z axis (Marya et al. (2014)). An example of the results produced by the *Given Imaging* RF system can be seen in Figure 2.30. More recent and accurate position estimation techniques are performed through magnetic sensors, despite being still experimental-only setups (Iakovidis et al. (2013)). With these hardware-based methods, not only the overall localization method cost increases, but there are also potential limitations in design and space, bio-incompatibility and even interference problems of the sensor with the device.

Following up on these issues, a series of vision-based methods were proposed. As an initial endeavor, structure from motion methods were developed (e.g. Fan et al. (2010) and Iakovidis et al.

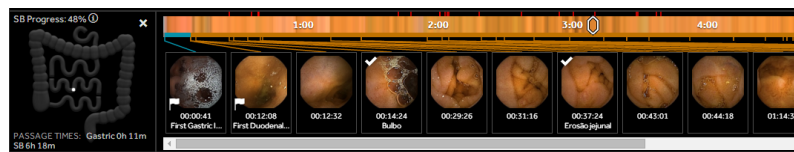


Figure 2.30: *Given Imaging PillCam Reader Software v9* showing the small bowel map, progress bar and thumbnails. The map is only shown when external sensors are used and after the first duodenal and first cecal images are manually marked landmarks

(2013)), although they were incapable of real-time processing. Visual simultaneous localization and mapping (VSLAM) in the medical field has been investigated in Mountney et al. (2006), being applied to endoscopic capsule localization in Grasa et al. (2009) and Grasa et al. (2014). Then, several VSLAM-based techniques were developed. In Lin et al. (2013) parallel tracking and mapping (PTAM) was advanced, allowing to build a denser 3D map than previous EKF-based SLAM methods. Another breakthrough VSLAM method was proposed in the form of ORB-SLAM, that was adjusted to endoscopic capsule localization in Mahmoud et al. (2016).

The exposed vision-based capsule localization algorithms rely on feature-based methods, searching for distinct features and tracking them across different frames to obtain an estimated position. Such methods generally perform poorly on endoscopic images, as these images show, most of the time, an absence of distinct landmarks. Also, with VSLAM-based algorithms the translation in the z axis direction is mathematically impossible to determine up to a scale with only one camera. Furthermore, most of the methods in present literature were developed with standard handheld endoscopy in mind. This can be problematic as endoscopic capsules display different characteristics: lower camera quality and resolution, space availability and limited energy source. (Turan et al. (2017b))

Bearing in mind all the issues posed by hardware-based and vision-based methods, a few methods that resort to deep learning techniques were recently presented. To our knowledge, the only work done in this field that makes usage of deep learning techniques was presented in Turan et al. (2017b), Turan et al. (2017a) and Turan et al. (2018).

Chapter 3

Visual Odometry Framework

3.1 Problem Characterization

The endoscopic capsule is already recognized as the main method to screen the entire small bowel. However, there are still some challenges to solve in order to release its full potential. In this work we will focus on the challenge of endoscopic capsule localization in the small intestine. Due to the 8 to 10 hours duration of the produced videos and the lack of landmarks throughout the small bowel, it is exceptionally difficult to make an estimation of the capsule position without additional information. So, it is important to develop methods that can provide this additional and valuable information to physicians performing a diagnosis and surgeons performing a surgery.

This localization information can be provided in terms of percentage of travelled distance inside the small bowel, which combined with the user medical experience can be very helpful when locating the capsule on the human body. Another option is to provide the capsule three dimensional position in each frame.

When trying to solve the endoscopic capsule localization problem, it is important to consider the following challenges:

1. Reduced frame rate can lead to no superimposition between consecutive frames.
2. There are large sequences of non-informative image frames that preclude any possibility of obtaining useful information. In this case, we consider an image non-informative when it contains a substantial amount of green residues or air bubbles. This phenomenon appears often in endoscopic images and they occlude any relevant feature.
3. It is not an easy task to obtain a capsule position groundtruth in an environment like the small intestine. Without a groundtruth, it is only possible to use unsupervised learning methods and supervised learning methods with artificially generated labels. Additionally, this makes the work validation much harder to obtain, as labeled data is necessary to compare against the obtained results.

4. Peristalsis, a radial contraction and relaxation of muscles that propagates via a wave-like motion, occurs throughout the gastrointestinal tract. This means that the movement detected might not correspond to real camera movement but to a change in the environment itself.

3.2 System Overview

In order to fulfill the physicians needs and to overcome the challenges posed by this type of dataset, we propose the framework presented in Figure 3.1. Our system requires four inputs: endoscopic images, the camera matrix, topographic landmarks and some capsule and small bowel information. The endoscopic images are the data our system will learn from and the data our system will be tested on. The topographic landmarks are used to determine in which frame the small bowel starts (duodenum) and in which frame it ends (ileocecal valve), as we are only interested in finding the capsule position in the small intestine. The intrinsic matrix is necessary for the system to obtain the characteristics of the camera used to record the endoscopic videos. The capsule information necessary is mainly the average and maximum capsule per frame displacements inside the small bowel. The small bowel information necessary is the minimum length, maximum length and average diameter.

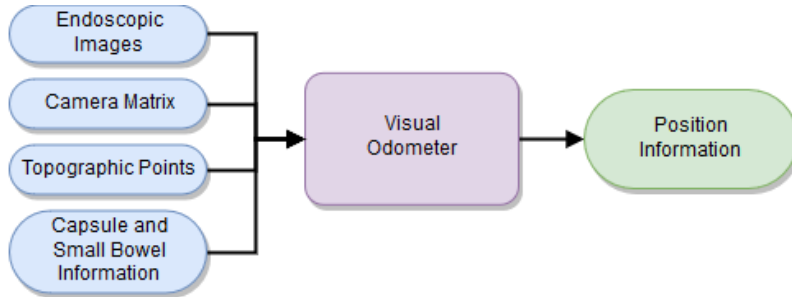


Figure 3.1: System framework overview

To obtain position information from these inputs we chose to apply a visual odometer, which has the great advantage of freeing the patient from uncomfortable additional external hardware. Two completely different types of deep learning based visual odometers were developed. One is based on the prediction of the homography between frames, performed by a HomographyNet, a convolutional neural network detailed in section 2.1.2.4 (DeTone et al. (2016)). Other is based on a depth and ego-estimation method (Zhou et al. (2017)), detailed in section 2.1.2.5. The first outputs a per frame displacement in percentage and the second a 3D capsule pose, a depth map for each frame and a small bowel 3D reconstruction.

Deep learning techniques have been setting new state-of-the-art results across different fields of computer vision, being this type of methods more robust to distortion effects, noise, occlusions, vignetting and deficiency of distinguishable features. It was mainly based on these properties and in the nature of the endoscopic datasets that we developed our methods.

3.2.1 Displacement Estimation - HomographyNet

In Figure 3.2 we can see the full framework that describes this method. What this visual odometry method seeks to accomplish is a per frame capsule displacement (in percentage) throughout the small bowel - similar to what is seen in Figure 2.30 - without any additional hardware components. In sum, it does not aim to be a tool that provides an exact three-dimensional localization, but one that can give useful information when combined with the physicians experience and knowledge.

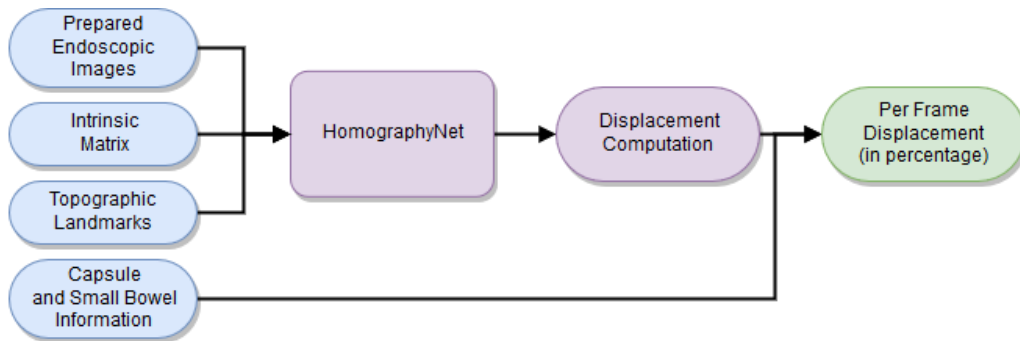


Figure 3.2: Displacement estimation framework

Firstly, the dataset needs to be prepared in a specific way in order to be fed to the HomographyNet. The HomographyNet outputs a 4-point homography prediction (section 2.1.1.4) that is then used to compute the displacement between frames. Then, this displacement information is fused with other capsule and small bowel information (average and maximum capsule per frame displacement inside the small bowel, small bowel minimum and maximum length and diameter) to provide a final per frame displacement in percentage. This fusion of information is necessary in order to detect abnormal displacement predictions and to fill in the gaps left by discarded non-informative frames.

The HomographyNet (detailed in 2.1.2.4) was chosen as the homography estimator for this framework because it performs better than previous state-of-the-art methods (as seen in DeTone et al. (2016)).

3.2.2 Depth and Pose Estimation - SfMLearner

The framework that combines pose and depth estimation, as well as a small bowel 3D reconstruction, can be seen in Figure 3.3. Here, we obtain a 3D localization with an end-to-end method that estimates depth and ego-motion and that can be trained in an unsupervised manner (Zhou et al. (2017)). The procedure consists in training two networks in the task of view-synthesis, which compels them to learn the transitional steps of image depth and camera 3D pose. Then, the 3D pose is refined with capsule and small bowel information (average and maximum capsule speeds inside the small bowel and small bowel diameter). This additional information allows us to detect and replace abnormal measures and to fill in the gaps of discarded non-informative frames. From the depth maps predicted for each frames is possible to extrapolate a 3D reconstruction.

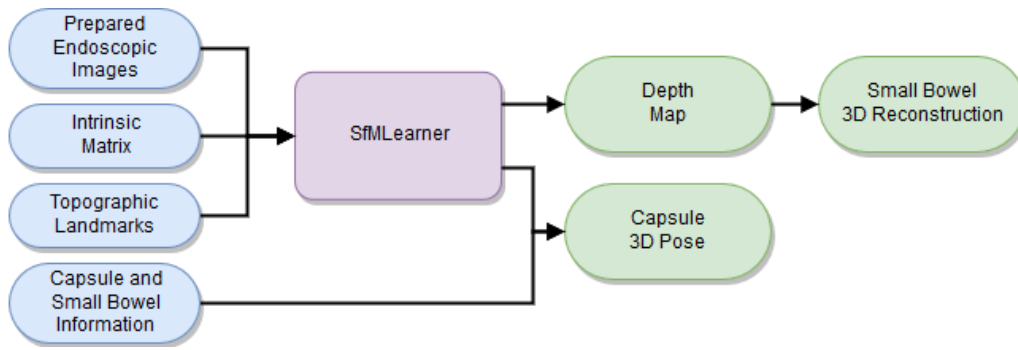


Figure 3.3: 3D pose and depth map estimation framework

There are two main reasons why this method is very well suited to solve an endoscopic capsule localization problem. First, the fact that it can be trained in a completely unsupervised way is truly important, due to the great level of difficulty and cost to obtain positioning labels in the gastrointestinal track. Second, an explainability network (see section 2.1.2.5) is trained to prevent phenomena which commonly occur on endoscopic videos, such as:

- Extremely common occlusions due to fluid or even solid particles.
- Non-Lambertian surfaces due to the reflections caused by the capsule illumination system.
- Gastrointestinal track deformation due to the non-rigid nature of the human organs.

3.3 Dataset

The anonymized dataset available to us consists of 449 endoscopic capsule videos, 338 from the PillCam SB2 endoscopic capsule and 111 from the PillCam SB3 endoscopic capsule. With that said, the dataset used in our work is only composed by the PillCam SB3 videos. This decision was based on the fact that this capsule is the latest iteration of PillCam SB capsules, which translates into an improved image quality with superior resolution and an increase in the frame rate. Furthermore, some of the videos captured by this capsule have a unitless displacement attached, which is based on landmarks marked by the physician (duodenum and ileocecal valve) and a radio-frequency (RF) sensor array. Even though this displacement can not be used as groundtruth to train a neural network (there is no easy way to export this information from the *PillCam Reader Software*¹), it can be used to validate our methods. The majority of the videos are medically annotated in terms of topographic landmarks (first duodenal, ileocecal valve and cecal image), as well as the identified lesions and abnormalities. Both capsules characteristics can be seen in Table 3.1 and a comparative example of their captured frames can be seen in Figure 3.4.

It is also important to notice that in order to transform the videos into a compatible format for our system, we converted them into image sequences using Sensarea (Bertolino (2014)).

¹<http://medtronicsolutions.medtronic.com/rapid-reader-9-0-download>



 <p>PillCam SB2</p>	 <p>PillCam SB3</p>
<ul style="list-style-type: none"> • $26mm \times 11mm$ • weight 3.7g • 7 – 8h battery life • resolution 256×256 • 2 fps • AOV 156° 	<ul style="list-style-type: none"> • $26.2mm \times 11.4mm$ • weight 3.0g • 8h or longer • resolution 320×320 • 2 – 6 fps AFR • AOV 156°

Table 3.1: PillCam SB2 and PillCam SB3 characteristics

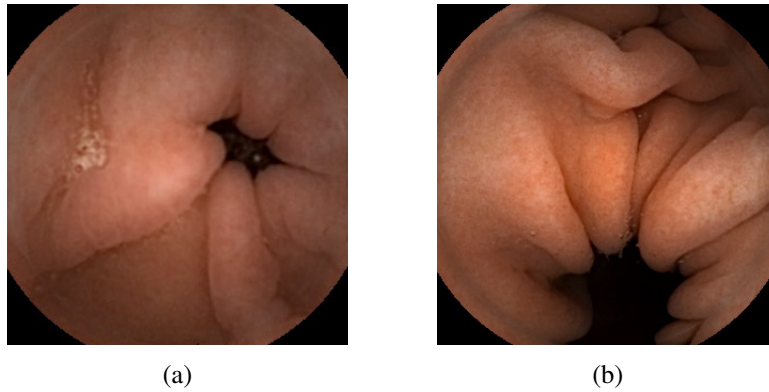


Figure 3.4: (a) PillCam SB2 captured frame example; (b) PillCam SB3 captured frame example;

We can see the traditional endoscopic dataset challenges (section 3.1) depicted in our dataset in Figures 3.5 and 3.6. Besides the typical endoscopic dataset characteristics, our dataset demonstrates another peculiarity that is important to note. The video software used to compile the raw data recorded by the endoscopic capsules (*PillCam Reader Software*) utilizes a smart video compiler, which greatly reduces the number of frames in the video, eliminating those it considers redundant. Those removed frames could potentially reduce the lack of superimposition between frames in some cases. To put this into perspective, the capsule frame rate can go as low as 0.7 after this technique is applied.

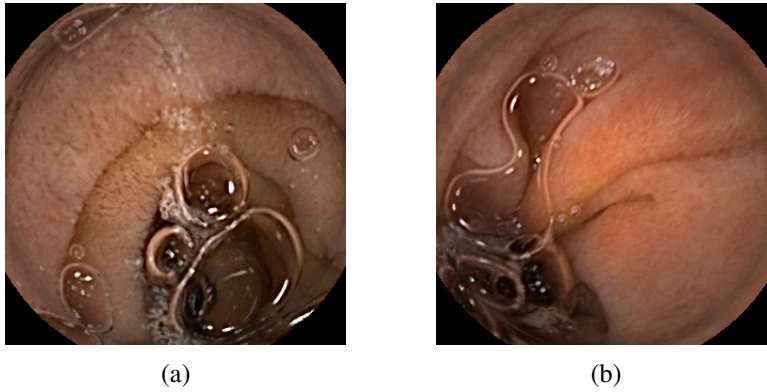


Figure 3.5: PillCam SB3 consecutive frames without superimposition.

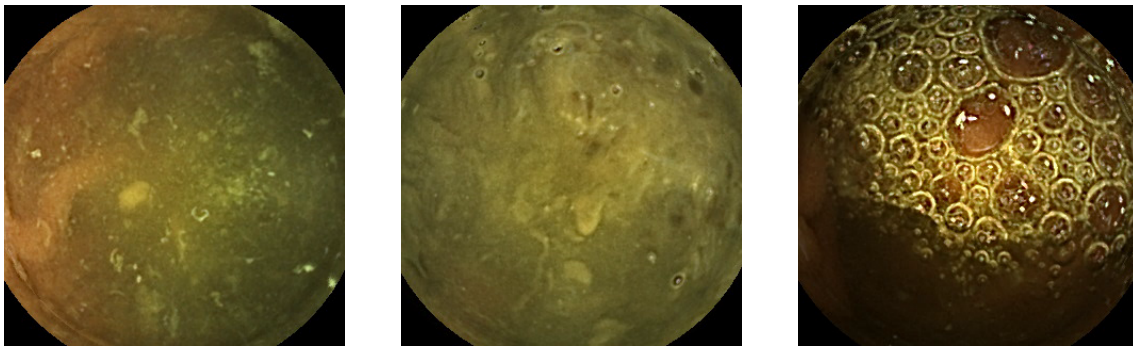


Figure 3.6: PillCam SB3 captured frames with noise. We consider noise the green residues and bubbles that occlude the features.

3.3.1 Treated Dataset

In order to solve the problem of non-informative images we propose two methods to create a new treated dataset. The first is a simpler approach, which is based on the dominant color of the dataset images and the other is a deep learning approach, which utilizes a convolutional neural network to predict if the images are informative or not.

The first method starts with the detection of the dominant color in RGB for each frame in the endoscopic dataset. Then, after getting the dominant color, we change the RGB results into the HSV color space, where it is easier to identify the characteristic green tones of most non-informative frames (e.g. Figure 3.6). Finally, the frame is considered non-informative and discarded if the hue value of the dominant color is between 21 and 80, a space interval which we found empirically to represent the green tone in non-informative images. With this threshold we found out that around 17% to 28% of each small bowel endoscopic video frames are considered non-informative. By visually inspecting the different videos analyzed, we concluded that the values should have been slightly higher. With that said, this method can still provide a good approximation of the amount of non-informative frames.

There are two big disadvantages when it comes to this method: the algorithm becomes computationally heavier (taking much more time to run) and its accuracy is not as good as desired

(e.g. it might detect mainly green frames that are still informative or fail to detect some frames with bubbles that do not show a green color but still occlude most features). This means that a dominant color only approach might not be sufficient.

In this sense, we propose the creation of a second alternative method to classify the usefulness of the frames. A convolutional neural network based on the MobileNet architecture, exposed in section 2.1.2.4), would be trained on a new dataset to predict if an endoscopic image belongs to the informative or non-informative class. As our dataset frames were not labeled as *informative* or *non-informative*, the first solution was used to create our new treated dataset.

3.3.2 Camera Calibration

In order to perform camera calibration, we applied *MATLAB Single Camera Calibrator App* (Figure 3.7). The procedure starts with the capture of multiple images of a checkerboard with the camera we wish to calibrate, in our case, the PillCam SB3 endoscopic capsule. We then need to give the size of the checkerboard squares (2 mm in our case) and finally select the threshold to eliminate the outliers based on the reprojection error. The obtained intrinsic matrix (K) (see section 2.1.1.1 for more details) and distortion coefficients are:

$$K = \begin{bmatrix} 160.7178 & 0 & 165.6875 \\ 0 & 161.2810 & 161.1860 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

$$\text{distortion coefficients} = \begin{bmatrix} -0.0218 & -0.2353 & 7.3532e-4 & -0.0069 & 0.0914 \end{bmatrix} \quad (3.2)$$

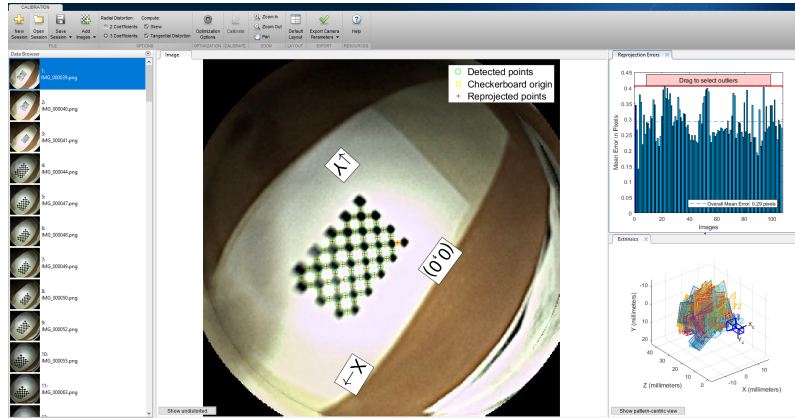


Figure 3.7: *MATLAB Single Camera Calibrator App* framework

Chapter 4

Results

4.1 Displacement Estimation - HomographyNet

As seen in the framework in Figure 3.2, there are two main processes to obtain the capsule displacement across the frames of an endoscopic capsule video. The first process, performed by the HomographyNet (DeTone et al. (2016)), is the prediction of the homography that describes the transformation between frames. Then, with the second process we translate the obtained homography into displacement information.

As we do not dispose of any dataset labeled with the homography transformations between frames, we need to create a synthetic dataset with artificial known distortions in order to train the networks.

4.1.1 Network Architectures

Our tests were conducted with two different models, the *Regression HomographyNet* and the *MobileNet-based HomographyNet*, both supervised learning methods that were explored in detail in section 2.1.2.4.

The *Regression HomographyNet* has an input shape of $128 \times 128 \times 2$, that is, two grayscale images stacked. It is composed of 8 convolutional layers followed by a ReLU activation function, batch normalization (convolutional block) and sometimes a max pooling layer. Finally, the result is flattened and put through two fully connected layers with dropout layers in between, to output a vector of size 8. This vector is composed by real numbers that represent the 4-point homography that describes the transformation between the input images. There are 8 values because, as detailed in section 2.1.1.4, the 4-point homography is composed by the offsets between each coordinate of each corner. A model summary can be seen in Table 4.1.

The *MobileNet-based HomographyNet* has the same two grayscale images stacked input as the *Regression HomographyNet*. It starts with a convolutional block composed by a convolutional layer with 32 filters with a 3×3 kernel and a stride of 2, followed by batch normalization and a ReLU activation function. Then, 13 depth-wise convolutional blocks take place, which consist of a depth-wise convolution, batch normalization, ReLU activation, a point-wise convolution, batch

normalization and another ReLU activation. Finally, it concatenates 4 convolutional layers, each with 2 filters with a 4×4 kernel and a stride of 1, then flattening the result to output a vector of size 8. This outputted vector is also composed by real numbers and also represents the 4-point homography between the input images. A model summary can be seen in Table 4.2.

Blocks/Layers	Filters	Kernel	Strides	Pool Size
Convolutional Block	64	3×3	1×1	
Convolutional Block	64	3×3	1×1	
Max Pooling Layer				2
Convolutional Block	64	3×3	1×1	
Convolutional Block	64	3×3	1×1	
Max Pooling Layer				2
Convolutional Block	128	3×3	1×1	
Convolutional Block	128	3×3	1×1	
Max Pooling Layer				2
Convolutional Block	128	3×3	1×1	
Convolutional Block	128	3×3	1×1	
Flatten				
Dropout				
Fully Connected = 1024				
ReLU				
Dropout				
Fully Connected = 8				
Total Parameters	34,195,336			

Table 4.1: HomographyNet Regression architecture

Blocks/Layers	Filters	Kernel	Strides
Convolutional Block	32	3×3	2×2
Depth-wise Convolutional Block	64 pw	3×3 dw	1×1 dw
Depth-wise Convolutional Block	128 pw	3×3 dw	2×2 dw
Depth-wise Convolutional Block	128 pw	3×3 dw	2×2 dw
Depth-wise Convolutional Block	256 pw	3×3 dw	2×2 dw
Depth-wise Convolutional Block	256 pw	3×3 dw	1×1 dw
Depth-wise Convolutional Block	512 pw	3×3 dw	2×2 dw
Depth-wise Convolutional Block	512 pw	3×3 dw	1×1 dw
Depth-wise Convolutional Block	512 pw	3×3 dw	1×1 dw
Depth-wise Convolutional Block	512 pw	3×3 dw	1×1 dw
Depth-wise Convolutional Block	512 pw	3×3 dw	1×1 dw
Depth-wise Convolutional Block	512 pw	3×3 dw	1×1 dw
Depth-wise Convolutional Block	1024 pw	3×3 dw	2×2 dw
Depth-wise Convolutional Block	1024 pw	3×3 dw	1×1 dw
Convolutional Layer	2	4×4	1×1
Convolutional Layer	2	4×4	1×1
Convolutional Layer	2	4×4	1×1
Convolutional Layer	2	4×4	1×1
Concatenate Layer			
Flatten Layer			
Total Parameters	3,359,656		

Table 4.2: MobileNet-based HomographyNet architecture

4.1.2 Synthetic Dataset

Both HomographyNet architectures we will implement for our first method are supervised learning methods, thus requiring labeled data. As we only possess an unlabeled dataset (see section 3.3), we need to artificially create our groundtruth. As we can create as many artificial transformations as we want, this will also allow us to have a seemingly infinite amount of data.

The networks were designed to take as input two 128×128 stacked grayscale images that are labeled with a homography matrix that relates them. These images are called patches because

they are a crop from 320×320 images. So, we created 3 labeled image pairs for each existing frame of the treated dataset described in section 3.3.1, which is the dataset used as basis. This synthetic dataset was created to mimic real transformations that occur between two consecutive frames, by randomly generating a great variety of transformations and by randomly introducing noise artifacts.

We will now describe the algorithm that allowed us to generate our synthetic dataset with an artificial 4-point homography (section 2.1.1.4) groundtruth for every created image pair. This algorithm takes a 320×320 RGB frame and outputs two stacked 128×128 grayscale patches.

1. Convert the original image to grayscale.
2. Crop a 128×128 square patch from the 320×320 grayscale image (blue square from Figure 4.1a). The patch vertexes coordinates are randomly generated between defined values, always maintaining the 128×128 size. This allows us to include different features for each image pair created from the same image.
3. Add a random perturbation to each of the patch vertexes coordinates (green square from Figure 4.1a). A perturbation is no more than the sum of a random value to each coordinate of each patch vertex.
4. Based on the original and perturbed points obtained from step 3, we calculate the homography matrix that maps the original image pixels to the perturbed image pixels with a perspective transformation function. We then proceed to compute the inverse of this matrix and warp the grayscale 320×320 image according to it in order to obtain a new 320 warped image.
5. Use the unperturbed generated vertexes to crop a new 128×128 patch of the newly warped 320 image (green square from Figure 4.1b).
6. Introduce different artifacts with random levels of intensity and with a certain probability, such as Gaussian blur with 30% of probability and increase or decrease in pixel intensity with 40% of probability (two possible examples can be seen in Figure 4.2a and Figure 4.2b).
7. Return the perturbed 128×128 patch stacked on top of the original 128×128 patch in order to create a $128 \times 128 \times 2$ image pair and return the offset between the perturbed patch vertexes and original patch vertexes coordinates. This offset will be our 4-point homography (see section 2.1.1.4) used as label.

The process described above is repeated 3 times for each image sample in our treated dataset (section 3.3.1), representing a threefold increase in our data size. This process was done for 75% of our treated dataset to create our train dataset and for the others 25% to create our test dataset, which translates to 7303680 training samples and 2711040 test samples.

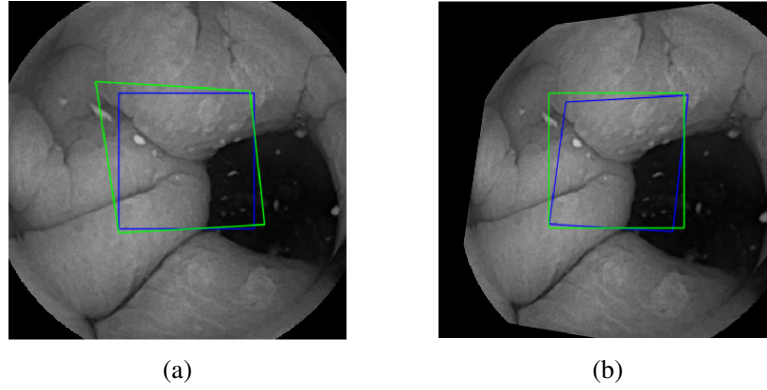


Figure 4.1: **(a)** Original patch and perturbed points; **(b)** Perturbed patch;

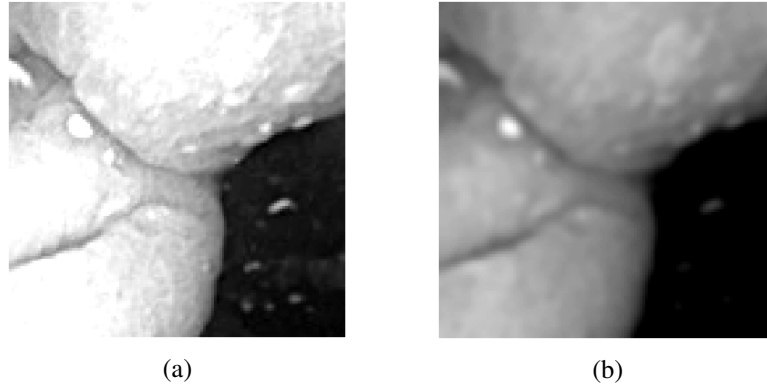


Figure 4.2: **(a)** Perturbed patch augmentation example 1; **(b)** Perturbed patch augmentation example 2;

4.1.3 Train and Synthetic Data Tests

The training process for both proposed architectures utilizes the synthetic train dataset described in section 4.1.2 and minimizes the loss function shown in equation 2.36 (as mentioned in DeTone et al. (2016)). Also, the following metric was used:

$$\text{Mean Corner Error (MCE)} : \frac{1}{n} \cdot \sum_{i=1}^n \sqrt{\sum_{k=1}^4 (p_{i,k} - q_{i,k})^2} \quad (4.1)$$

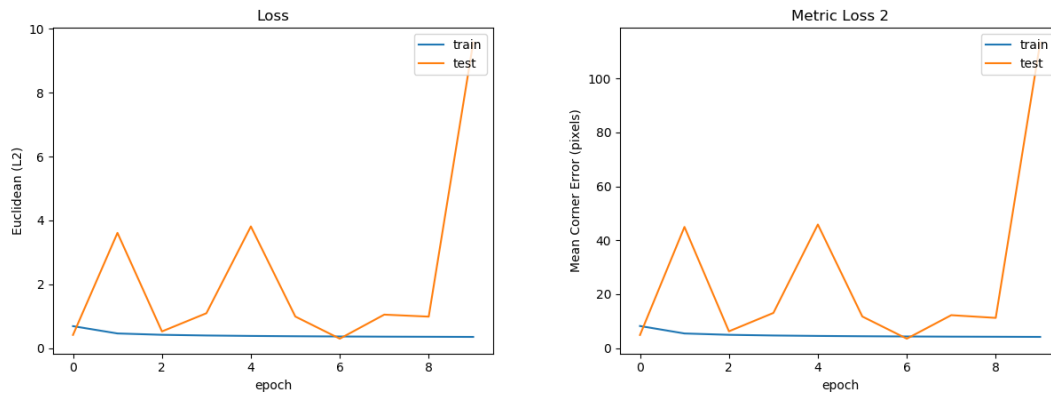
being n the number of image pairs, $p_{i,k}$ the predicted values and $q_{i,k}$ the true values for each corner of each image pair. This metric is an intuitive way to verify the network success because, as the name suggests, it represents the average of the error of the predicted corner offsets of each image, in pixels.

The final hyper-parameters chosen to train the networks were:

- Batch size of 64
- 9 epochs
- *Stochastic Gradient Descent* optimizer with a learning rate of 0.005 and a momentum of 0.9

In Figures 4.3 and 4.4 we can see the results obtained in the train dataset and test dataset along the 9 training epochs. We can conclude that the regression HomographyNet performs considerably worse, showing a lot of difficulty to minimize the error, especially in our test dataset. Empirically, we found that although the MobileNet-based model always presented better results, the regression model converged correctly before we added noise artifacts (in form of pixel intensity changes and Gaussian blur) to our synthetic dataset. The MobileNet-based model not only performs better, but also presents a much smaller size than its counterpart, being faster to train and to make predictions.

So, the best achieved results were obtained after 9 epochs of training the MobileNet-based model, with an average loss of 0.0029 and a mean corner error of 1.3590 pixels on the test dataset. Taking into consideration the input image sizes of 320×320 , a mean corner error of 1.3590 pixels is practically negligible.



[Graphical representation of the regression HomographyNet results]

Figure 4.3: Graphical representation of the regression HomographyNet results. From left to right we have represented the *Euclidean L2* loss and the *Mean Corner Error (MCE)* metric in pixels. In blue we have the results in the training set and in orange in the test set.

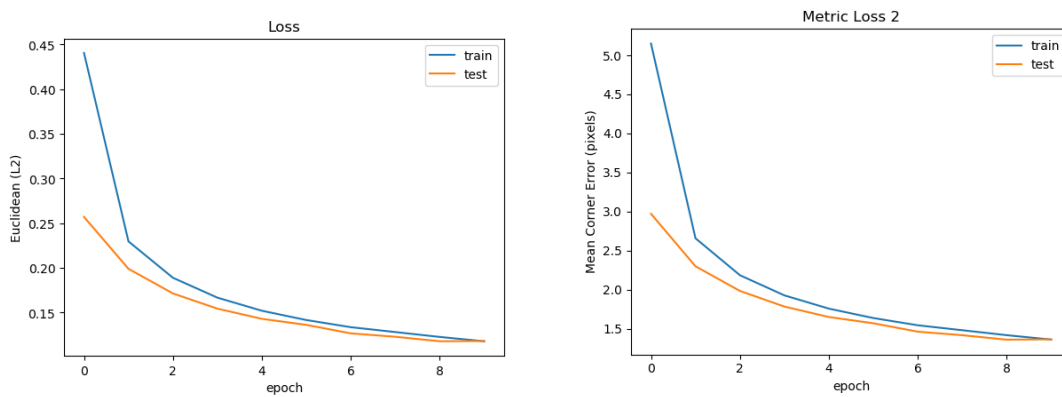


Figure 4.4: Graphical representation of the MobileNet-based HomographyNet results. From left to right we have represented the *Euclidean L2* loss and the *Mean Corner Error (MCE)* metric in pixels. In blue we have the results in the training set and in orange in the test set.

A visual representation of the accuracy of the obtained results can be seen in Figure 4.5.

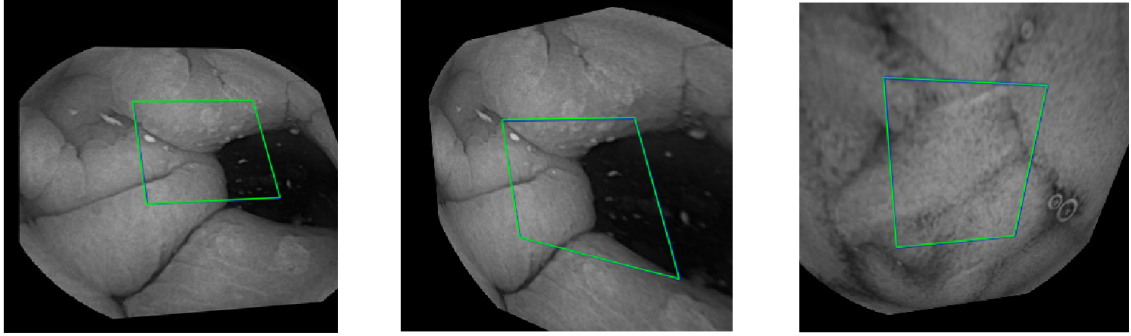


Figure 4.5: Multiple examples showing the predictions accuracy. The blue line represents the distorted patch and the green line the predicted patch. The blue line is barely visible as the prediction and groundtruth are almost superimposed.

4.1.4 Real Data Tests

In this section we will display and discuss the results obtained when using the synthetically trained MobileNet-based model (section 4.1.3) to make predictions on real data only (from the treated dataset described in 3.3.1). Testing this model on real data means evaluating the 4-point homography prediction for two consecutive endoscopic video frames. Here, the main challenge is to accurately evaluate the outcome, as there is no available groundtruth to use as a baseline. So, the only way to verify the precision of our work is to warp the first frame in an image pair according to the predicted 4-point homography between them, then comparing their similarity.

As we are now using full images instead of centered patches (where the distortion is minimum), it is important to address the big distortion (normal in any wide-angle lens) that occurs at the borders of our images. In that sense, we use the *intrinsic matrix* (K) and the *distortion coefficients* of our camera (obtained in section 3.3.2) in order to undistort our images. We can see in Figure 4.6 that this further improves the network predictions. This improvement can probably be explained by the added spatial consistency across the image that the undistortion provides.

Although we obtained great results predicting the homography in synthetic data by feeding only two 128×128 cropped patches to the network, that is not as efficient when testing on real data. This is expected if we admit that in the first case the transformation is always controlled, almost assuring that there will be a big enough superimposed area on both patches, as opposed to a real data test, where there might be a great enough camera shift that the overlapped areas might be too far apart. So, instead of feeding a pair of cropped 128×128 patches to the network, we resize each 320×320 frame from an image pair to a smaller size of 128×128 . This way, when stacking two consecutive frames, the $(128 \times 128 \times 2)$ format required by the network is still maintained, while maximizing the number of features that can be detected. In Figure 4.7, we can see a comparison between both approaches, where the superiority of the resizing approach is clear.

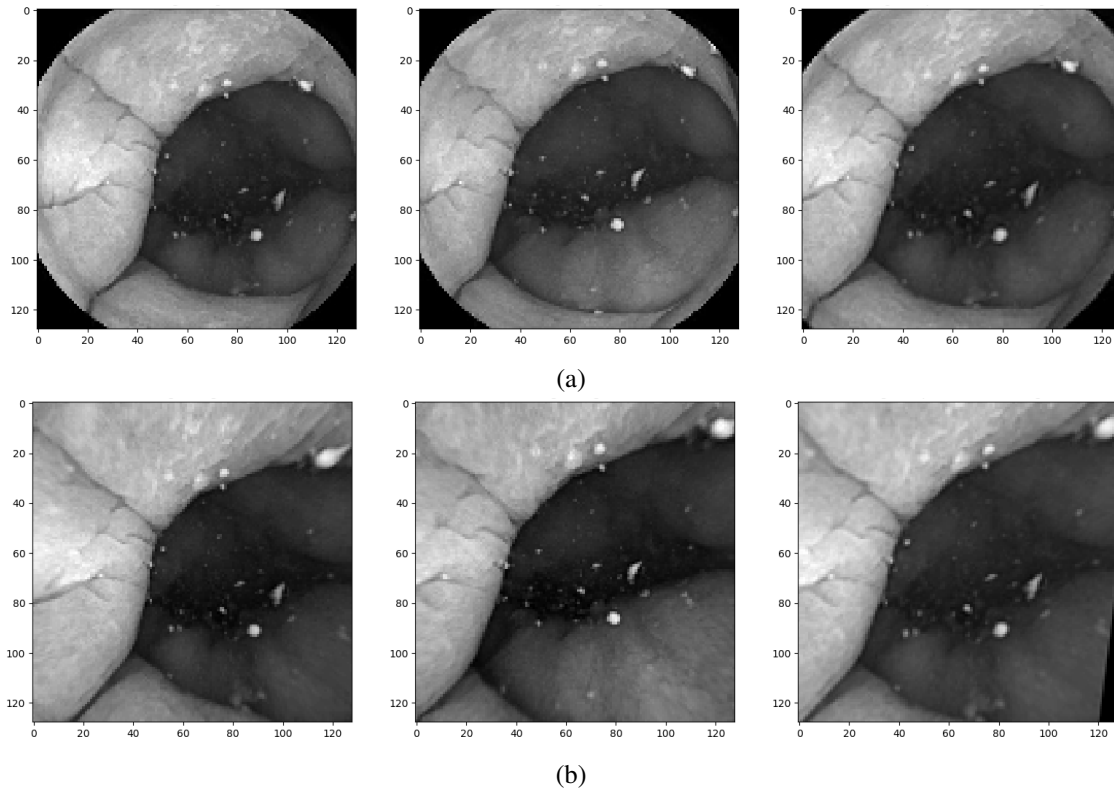


Figure 4.6: Comparison between predictions with distorted and undistorted images. From left to right: first frame, second frame and first frame warped to the second frame perspective. **(a)** Prediction using regular consecutive frames; **(b)** Prediction using undistorted consecutive frames;

In Figure 4.7 we also display the transformation obtained when using a SIFT + RANSAC approach (section 2.1.1.3), a classical computer vision feature-based approach. As endoscopic datasets lack on distinct features, feature-based methods generally perform poorly. SIFT poor performance can be seen in Figure 4.7, only being able to find 7 inlier feature matches between both frames. This shows how efficient our deep-learning method is on datasets of this nature.

In Figure 4.8 we can see multiple examples where the first frame in a pair is warped to the second frame perspective according to the predicted homography. In order to interpret this results we need to inspect the scale on the three images given that compose each example. In the first example it is especially noticeable how the bottom of the image is transformed in order to match the second frame view. On the second and third examples we can clearly notice that the predictions reflect the overall *zoom-in* that occurs between the first and second frames of the image pairs. In the last example, even though a more complex situation is depicted, we can clearly see the shift in the y axis and the slight *zoom-out* being correctly reflected in the prediction.

We concluded that our method performs well in real data, especially when compared with feature-based approaches like SIFT. With that said, this method alone is still flawed (Figure 4.9) and, a failed prediction will probably be caused by one of these issues:

1. Lack of distinct features caused by a small superimposed are between frames.

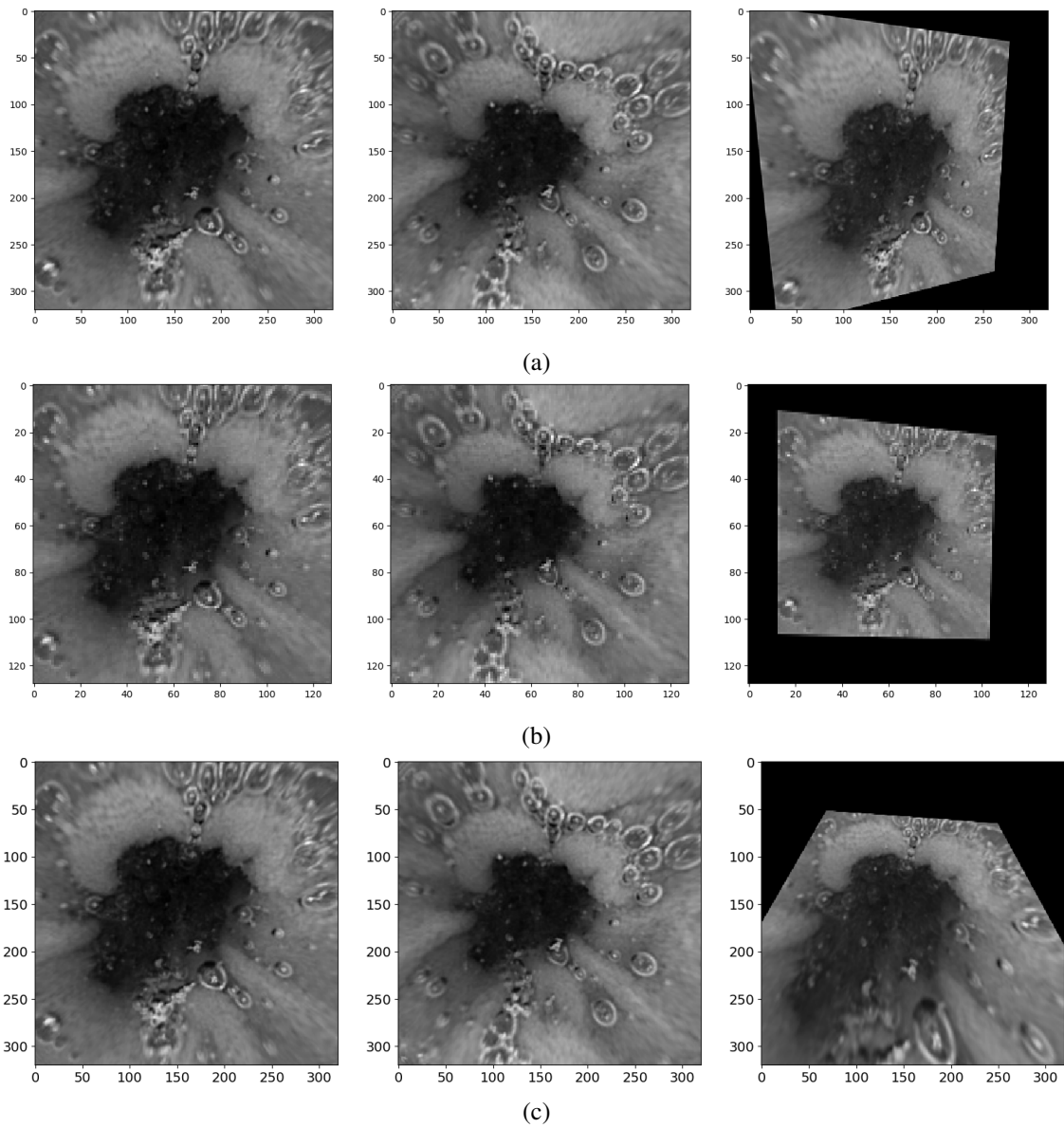


Figure 4.7: Comparison between three alternative homography estimation approaches. From left to right: first frame, second frame and first frame warped to the second frame perspective. **(a)** HomographyNet with centered single-patch approach; **(b)** HomographyNet with image resize approach; **(c)** SIFT + RANSAC approach.

2. Lack of context. There are some transformations trivial to a human inspector that might be a difficult task to our network, which might be explained by the absence of memory in our model.

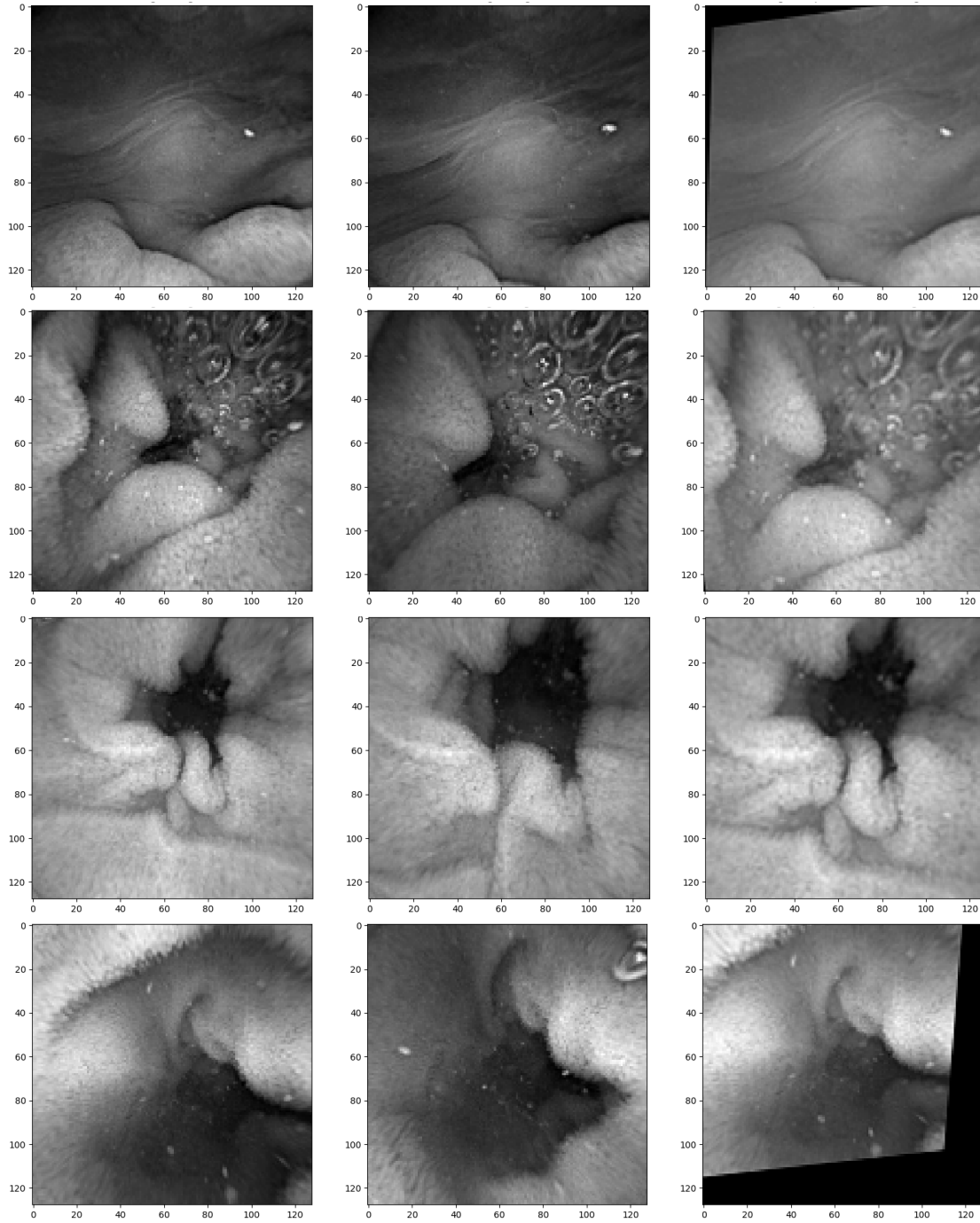


Figure 4.8: Multiple examples showing the homography predictions in real data. From left to right: first frame, second frame and first frame warped to the second frame perspective.

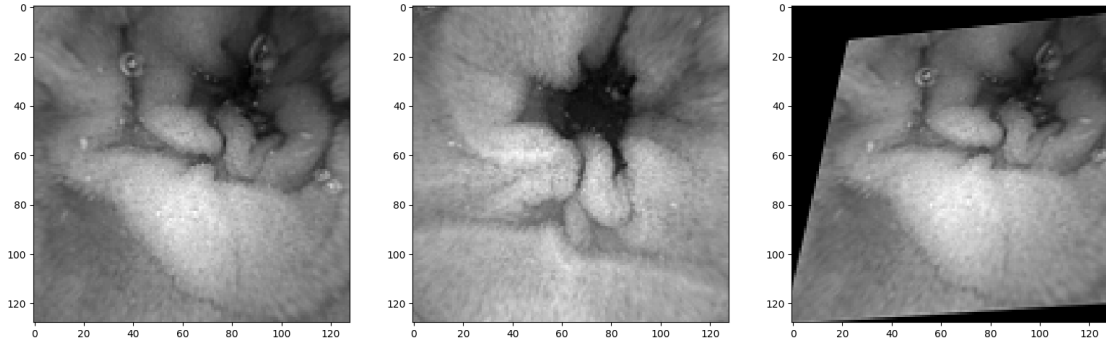


Figure 4.9: Unsuccessful homography prediction between two consecutive frames

4.1.5 Displacement Computation

The overview framework of the algorithm used to compute a displacement between a frame pair is depicted in Figure 4.10. The dataset used as basis was the treated dataset detailed in section 3.3.1.

We start by verifying if both frames we are analyzing are informative. If they are, we start the frames processing. Each frame is converted to grayscale, undistorted (with the values obtained in section 3.3.2)) and resized from 320×320 to 128×128 . Then, the processed images of the frame pair are stacked together in order to be fed into the MobileNet-based HomographyNet.

The HomographyNet then predicts a 4-point homography (detailed in section 2.1.1.4), which is then transformed into a 3×3 homography matrix that relates both frames and describes the camera movement between them. Now we can multiply the 2D coordinates of the previous camera position by the obtained homography in order to get the current camera position. The capsule positions based on this coordinates are depicted as black dots in Figure 4.11.

As these coordinates are obtained in pixels, we need to scale them to meters. This is done by multiplying them by $\frac{0.023}{128}$, being 128 the testing images size and 0.023 an average diameter of the small intestine diameter in adults (Haworth et al. (1967)).

We then calculate the uni-dimensional displacement (d_x in Figure 4.11) based on the computed camera coordinates (p and p' in Figure 4.11). The displacement along the x axis is the only displacement that is interesting for a medical diagnostic. So, we can compute this displacement as follows:

$$d_x = p'_x - p_x \quad (4.2)$$

We can disregard the movement on the z axis (up and down) without losing much information because the small bowel is a tube of small diameter (an average diameter of 0.023 meters in adults Haworth et al. (1967)), limiting the amount the capsule can move along this axis.

Then, in order to filter nonsensical displacement computation, we define a maximum threshold of $4.7565mm$. According to Bao et al. (2014), for a capsule that records at a frame rate of 2, $10mm/s$ is the speed limit to detect common features between consecutive frames. With an average

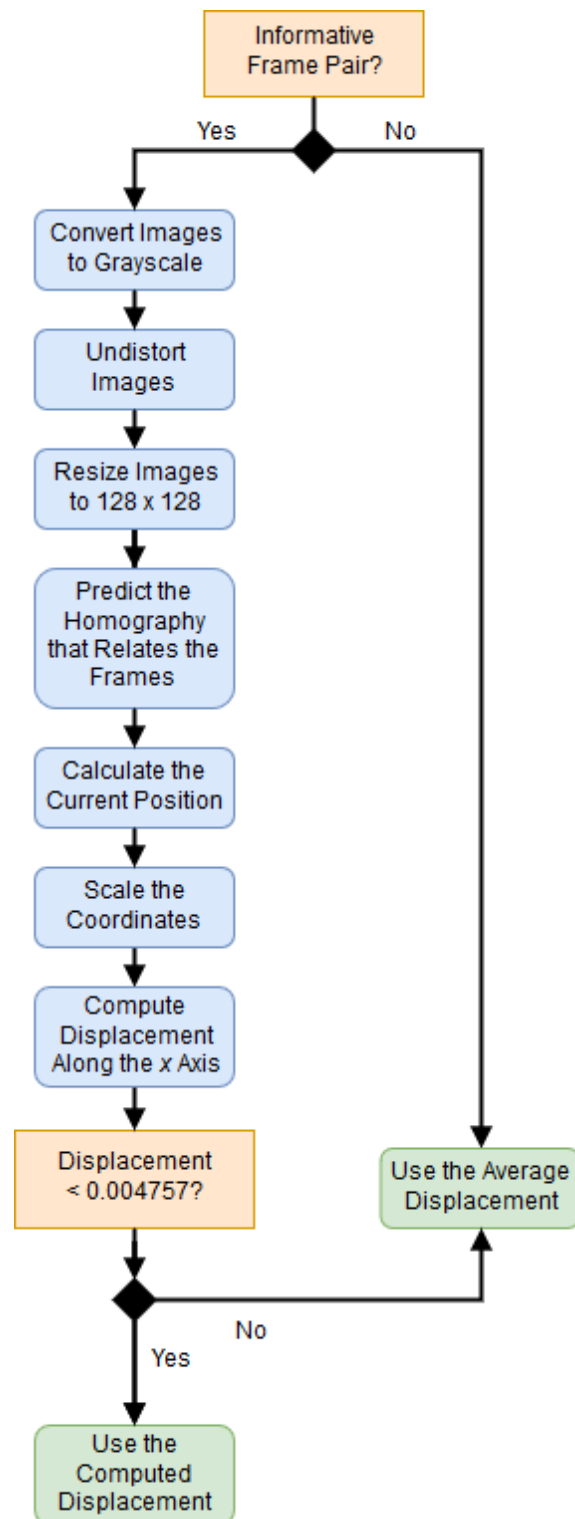


Figure 4.10: Displacement computation framework

frame rate of 0.7, the speed limit for our capsule is 3.5mm/s . With a frame rate of 0.7 we get that

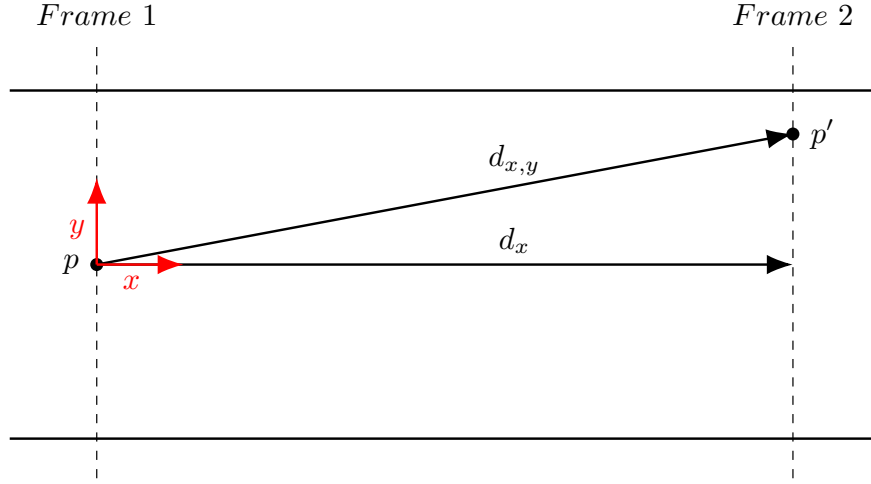


Figure 4.11: Top view for the capsule motion inside the small intestine

a frame is captured every 1.429 seconds:

$$\frac{1}{0.7} \approx 1.429 \text{ s} \quad (4.3)$$

So, in order to detect enough features to make a prediction, the maximum displacement per frame in our endoscopic videos should be:

$$3.5 \times 1.429 = 5.0015 \text{ mm} \quad (4.4)$$

If we detect that a displacement estimation is nonsensical, we need to make a new displacement approximation. Although we might think that using the last successful displacement estimation would be a good approximation, this is actually not true, especially for long non-informative sequences that commonly appear in our endoscopic videos. Keeping that in mind, we propose the use of an average per frame displacement of the endoscopic capsule in the small bowel. So, assuming that the capsule has an average speed of 0.48 mm/s (Bao et al. (2014)) and that our capsule captures a frame every 1.429 seconds (equation 4.4), we can calculate our average displacement per frame as follows:

$$0.48 \times 1.429 \approx 0.686 \text{ mm} \quad (4.5)$$

Displacement Results

The dataset used as base to test our results will be an altered version of the treated dataset detailed in 3.3.1, that only contains the frames that depict the small bowel passages. In order to perform this frame selection we used the duodenum and ileocecal valve topographic annotations (the beginning and ending of the small intestine) to crop the videos (section 3.3).

We will start by testing our method on small controlled frame sequences that only contain clean frames with plenty of overlapping area between them. With small frame sequences it is possible to visualize the state of the capsule progression and, in this way, compare it to our displacement results.

In Figure 4.12, we present a frame sequence where the capsule is clearly moving backwards. If we apply our algorithm to this frame sequence, the result also shows a backward motion (Figure 4.13). Even though the distance travelled from the first to the second frame appears to be the largest when looking at the frame sequence, it is difficult to justify a so accentuated difference when compared with the rest of travelled distances. As we have no precise scale reference it is not easy to evaluate it, but taking into consideration the scale of the small bowel environment, a travelled distance of $7mm$ along the frame sequence in Figure 4.12 seems correct.

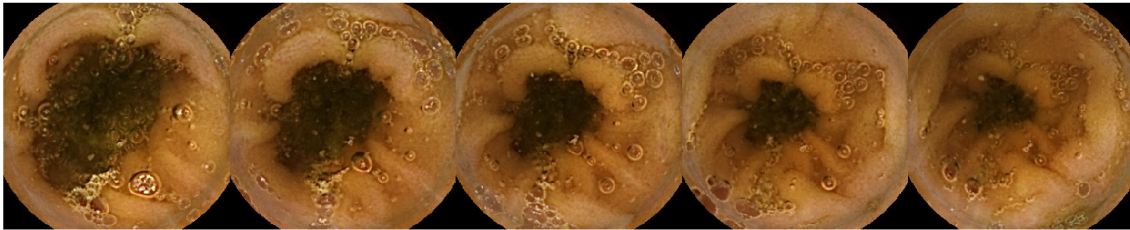


Figure 4.12: Frame sequence where the PillCam SB3 capsule is going backwards. The chronological order is from left to right and from top to bottom.

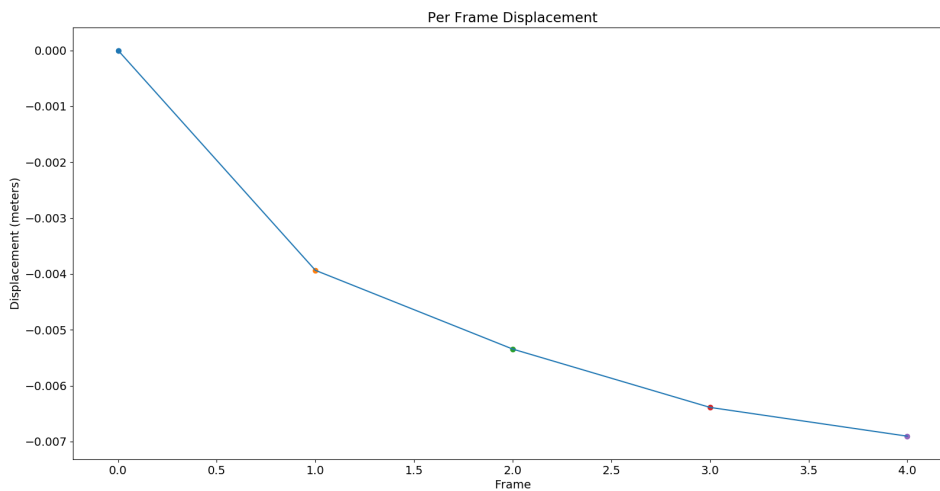


Figure 4.13: Backward displacement results from running our method on the frame sequence in Figure 4.12

In Figure 4.14, we have a frame sequence where the endoscopic capsule describes a forward motion. Applying our algorithm to this sequence will also show a forward motion (Figure 4.15). Once again, the distance travelled between frames seems correct, except for the distance travelled between the last two frames. Here, we can see that our algorithm detects that the displacement was

too high, marking it as a faulty detection and replacing it with the average capsule displacement of $0.686mm$ (equation 4.6).

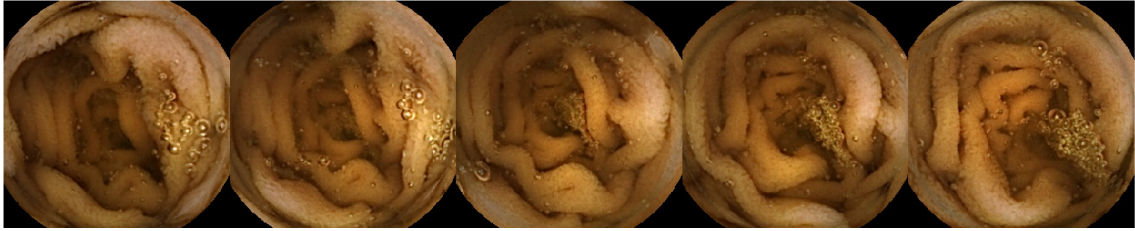


Figure 4.14: Frame sequence where the PillCam SB3 capsule is going forward. The chronological order is from left to right and from top to bottom.

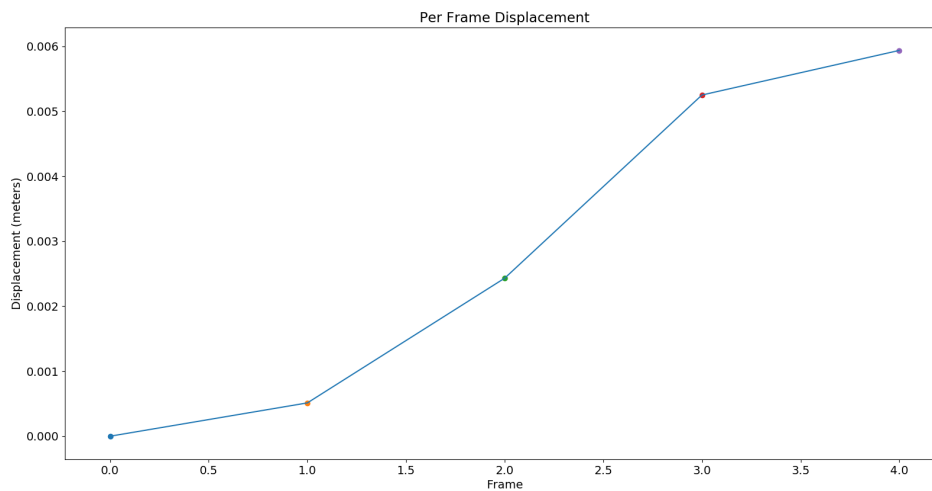


Figure 4.15: Forward displacement results from running our method on the frame sequence in Figure 4.14

We will now move on to validate our method on full small bowel endoscopic videos. As stated in section 3.3, even though we can not use our per frame displacement percentage labels to easily train a neural network, we can still use them to validate our results. We will only show our results in percentage because it is the only type of displacement that a physician needs in order to make a capsule position assessment, being easier to estimate a capsule position by interpreting a displacement result in percentage than in meters.

In Figure 4.16 we display the results (in percentage) obtained by running our algorithm on two different endoscopic capsule videos that capture the entire small bowel. While computing the displacement in Figure 4.16a video, 9933 out of 17019 displacement measures were discarded ($\approx 58.36\%$), whereas in Figure 4.16b video, 9095 out of 16659 were discarded ($\approx 54.95\%$). These measures were discarded as defined in the displacement computation framework (Figure 4.10).

To validate our results we compared them with the displacement (in percentage) obtained with *Given Imaging* proprietary software and hardware (Figure 4.17). As *Given Imaging* displays its

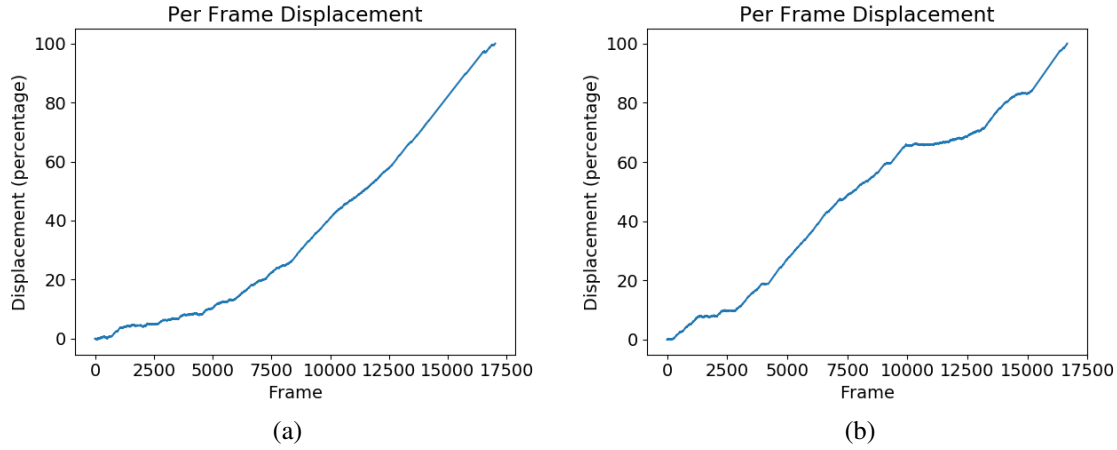


Figure 4.16: Per frame displacement (in percentage) in two examples of small bowel PillCam SB3 endoscopic capsule videos

results in a lower resolution of 101 (0% to 100%), we need to sub-sample our results for the methods resolutions to match. Then, considering the *Given Imaging* results as the reference value, we calculated the *Mean Absolute Error* (MAE) and its *Standard Deviation* (SD) for the computed displacements in Figure 4.16a video:

$$error = 5.00 \pm 2.65\% \quad (4.6)$$

and Figure 4.16b video:

$$error = 3.16 \pm 2.08\% \quad (4.7)$$

Considering the challenges of the endoscopic capsule localization problem and the accuracy of the methods currently used by physicians to determine the capsule position, we can conclude that the obtained errors are within the expected values. The bigger error in the first video might be explained by the bigger percentage of discarded measurements (which are replaced by an average displacement) and by undetected incorrect homography predictions. Ultimately, it is possible to conclude that our displacement computation method can provide truly useful information for a physician performing a diagnostic or even a surgery, without resorting to external equipment that would further disturb the patient. With that being said, there is still room for improvement, which will be address in section 5.2.

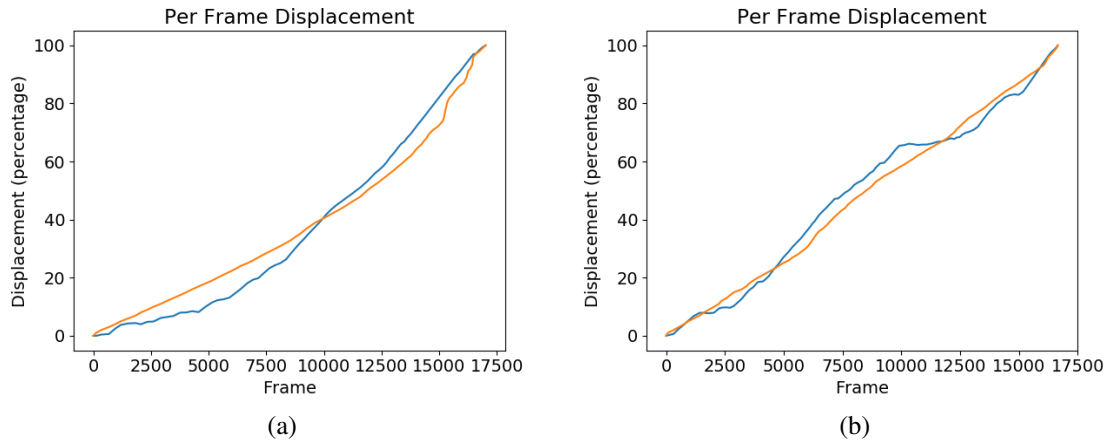


Figure 4.17: Comparison between our sub-sampled results and the results obtained in *Given Imaging* software. Our results are depicted in blue and the *Given Imaging* results in orange.

4.2 Depth and Pose Estimation - SfMLearner

As seen in the framework in Figure 3.3 and explained in section 3.2.2, this method is an end-to-end approach to the problem of the capsule position estimation. With this method we intend to estimate not only a three-dimensional position, but also a depth map for each frame. The 3D reconstruction, although proposed in our original framework, is not yet implemented.

This method consists in a model (detailed in section 2.1.2.5) composed by three neural networks that need to be trained in an unsupervised way. In order to train this model the data needs to be formatted in a specific manner. For this reason a new dataset was created.

4.2.1 SfMLearner Dataset

The creation of this new dataset uses as basis the original dataset detailed in section 3.3. The treated dataset detailed in section 3.3.1 (without non-informative images) was not utilized because we have not yet implemented a replacement for the discarded images. If in a one-dimensional environment, as in section 4.1, we can use an average displacement to replace discarded predictions, in a three-dimensional environment this problem becomes much more complex as we also need to deal with the capsule 3D orientation.

We started by cropping the videos to only depict the passages in the small bowel, by selecting the frames between the duodenum and the ileocecal valve. This is done because we are only interested to predict the capsule position inside the small bowel. The inclusion of the rest of the video in the training process could be problematic due to this reasons:

- Before the duodenum the capsule travels much faster, so consecutive frames often do not have overlapped areas.
- After the ileocecal valve, most of the image frames are usually non-informative due to green residues.

Then, we divide all the data into snippets of two or more consecutive frames (Figure 4.18), in a way that between snippets there is always a shift of one frame. That is, if the first snippet is composed by frames 1, 2 and 3, this means the second one is composed by frames 2, 3 and 4 and so on. Each snippet is composed by a target view frame and two or more nearby view frames (see details in section 3.2.2).

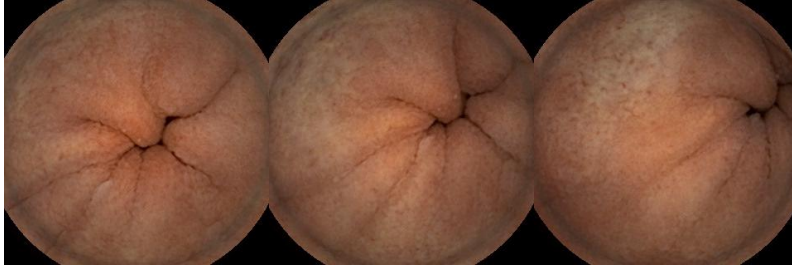


Figure 4.18: Example of three frames generated snippet

For each snippet, a text file with the camera *intrinsic matrix* (K) (section 2.1.1.1) is also created. The calibration process necessary to obtain the matrix K was already described previously in section 3.3.2. Finally, two other text files are created, one containing the directory paths for the snippets destined to training and the others destined to testing. These are randomly divided and represent 89% and 11% of the dataset, respectively.

4.2.2 Training

The hyper-parameters chosen for training this model were:

- Snippet length of 3
- Adam optimizer with a learning Rate of 0.0002 and momentum of 0.9
- Smoothness weight of 0.2
- Explainability regularization weight of 1.0
- 6 epochs

We experimented with various number of frames per snippet, but we could only train our network with a combination of 3 frames, which means that the information might be too disparate between five sequential frames.

According to its function of preventing phenomena such as occlusions (more details in section 3.2.2) and due to the nature of our dataset, the explainability is set to a high value of 1. The rest of the hyper-parameters were found in a purely empirical way.

It should be noted that while loading the images for training, a process of data augmentation takes place, consisting of random cropping and random scaling the images, which also implies a intrinsic matrix scaling in order to maintain the consistency. This will increase the effective size

of the training data and will help the network learn to cope with all the different distortions that can occur.

The total loss throughout the training process can be seen in Figure 4.19. As we can notice, the total loss does not actually converge. This is due to the fact that the photometric loss is not directly correlated with depth map quality. So, in order to evaluate how the training is progressing, *TensorBoard* (Abadi et al. (2016)) can be used to visualize the quality of depth maps being predicted during the process.

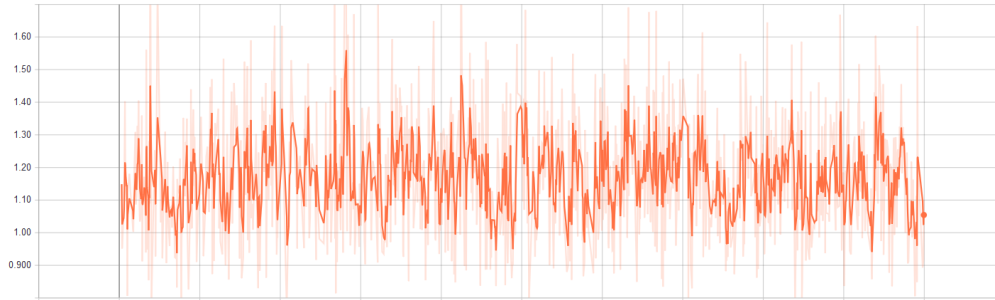


Figure 4.19: Total loss (equation 2.52) across the 200000 training iterations, equivalent to 6 epochs

4.2.3 3D Pose and Depth Results

In this section we will display the predictions obtained with this method on our dataset. We will show examples of depth predictions on several images and then present pose predictions on short frame sequences. Also, the quality of the results and our thoughts on how to improve them will be discussed. With that said, we have no available depth or camera poses ground-truths, so we can only validate our results in a manual and visual way.

In Figures 4.20a and 4.20b we display the depth results obtained for the frame sequences in Figures 4.12 and 4.14, respectively. We can verify that the depth results were not satisfactory, not depicting the images depth in an accurate way. The obtained results of the estimated 3D poses for the same sequences (Figure 4.21) were also not satisfactory, being difficult to justify the accentuated curvatures that occur on both examples. With that in mind, we can at least conclude that, in the shown examples, the predicted capsule motion directions are correct, as the capsule is predicted to be moving backwards for the sequence in Figure 4.12 and forwards for the sequence in Figure 4.14.

As this method is not yet prepared to deal with non-informative images, we will not apply it to full small bowel endoscopic videos. Even though our implementation was not successful, unsupervised methods like this one might be the future for the endoscopic capsule localization problem. They show a lot of potential to handle the challenges posed by a dataset of this nature and can help overcome the lack of labeled data in this field. Our thoughts on how to improve the obtained results will be exposed in 5.2.

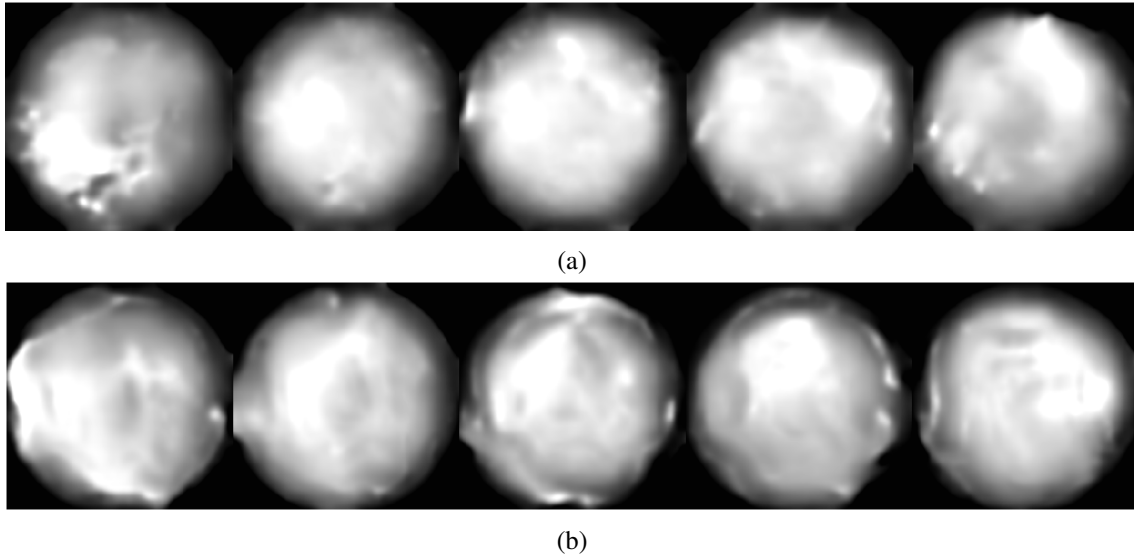


Figure 4.20: **(a)** Depth result from Figure 4.12 frame sequence; **(b)** Depth result from Figure 4.14 frame sequence;

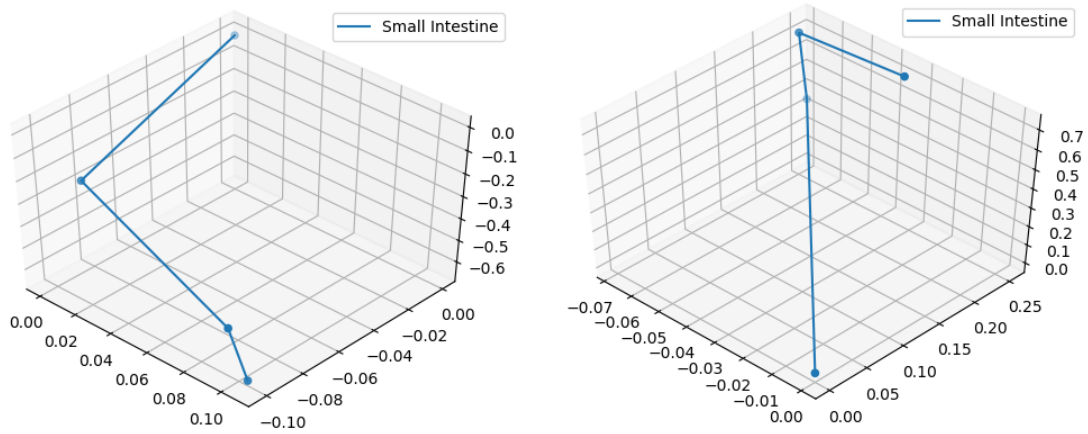


Figure 4.21: **(a)** 3D pose estimation for the frame sequence in figure 4.12; **(b)** 3D pose estimation for the frame sequence in figure 4.14;

Chapter 5

Conclusions and Future Work

The capsule endoscopy is now the main method for the diagnostic of the entire small bowel, an unreachable area by the traditional handled endoscopy and colonoscopy. Although it is crucial to identify abnormalities localization, easing follow-up examinations and surgical interventions, these capsules do not provide a localization. In order to track them along the small intestine, commercially available hardware methods such as radio-frequency sensor arrays can be used, although they entail several drawbacks. Developed software solutions show great potential, but still require improvements.

In this dissertation we developed two novel visual odometry solutions for the endoscopic capsule localization problem. The first method presents a solution where the displacement per frame is calculated (in percentage) throughout the small bowel. On the other hand, the second method intends to predict three-dimensional camera pose coordinates in each frame, as well as the corresponding depth map.

Both methods utilize deep learning techniques as their basis. The first one utilizes a convolutional neural network (DeTone et al. (2016)) in order to predict the homography that relates two consecutive frames, then translating it into displacement. The second one is an unsupervised and end-to-end approach that leverages from the task of *novel view synthesis* to obtain the intermediate steps of 3D pose and depth (Zhou et al. (2017)).

In order to further refine our results, we propose two different methods to classify frames as *informative* or *non-informative*. The first method was implemented and tested, being based on the frames dominant color. The second was only theorized, being based on a MobileNet-based convolutional neural network.

5.1 Objectives Satisfaction

The main objective for this dissertation was to create a visual odometry tool for the problem of endoscopic capsule localization in the small bowel. To accomplish this, we proposed and implemented two different frameworks based on deep-learning techniques: one that estimates the capsule displacement and other that estimates its three-dimensional pose. Also, methods to

deal with non-informative frames and nonsensical displacements were developed with success, although there is still space for improvement.

With the testing done to the displacement computation method (section 4.1), we can conclude that our method can provide vital capsule position information for the physicians performing a diagnostic or a surgery. The results obtained in synthetic datasets validated that the HomographyNet could successfully predict homography transformations between endoscopic capsule images, with a small average mean corner error of 1.3590 pixels. Then, with the predicted homographies on a real dataset, we visually validated that it could deal with real transformations between frames. To test the displacement computation, we first experimented with controlled frame sequences, hand-picking a series of informative frames with considerable overlapping areas between them. The results corresponded with the visually perceived motion in the frame sequences, validating our displacement computation. Then, we successfully validated the usefulness of our method on a clinical environment by testing it on full small bowel frame sequences and comparing our results to those obtained via radio-frequency methods. For the two tested videos we obtained mean absolute errors of $5.00 \pm 2.65\%$ and $3.16 \pm 2.08\%$, values within the expected when considering the currently used methods and the outlines of the proposed challenge.

With the tests conducted with the depth and ego-estimation method (section 4.2), we concluded that this implementation does not perform in a satisfactory way when applied to our problem, failing to predict accurate depth and camera poses on small frame sequences. We did not test with full small bowel frame sequences, as the implementation was not equipped to deal with non-informative frames yet. Even though the implementation failed to perform accurately, we are still convinced of its potential to solve this problem in the future, as its unsupervised nature and explainability mask are tools that, theoretically, are ideal to overcome some of the biggest challenges posed by a dataset of this nature.

When it comes to general knowledge related to the problem, we gathered a research summary of the existing endoscopic capsule localization techniques. Not only that, but we also documented the common difficulties researchers faced when using a similar dataset. Furthermore, a summarized research on the state-of-the-art motion estimation algorithms was developed.

To be done, was the implementation of a process to deal with non-informative frames and nonsensical displacements for the depth and ego estimation method and to further validate our frameworks on a labeled endoscopic dataset.

We consider that this dissertation has positively contributed to the problem of endoscopic capsules localization by studying and making use of state-of-the-art deep learning techniques in order to develop and explore new visual odometry methods. In our opinion, the application of such techniques show great potential and can definitely be disruptive in this field. Naturally, being this project still at its infancy, there is still a lot of fine-tuning and experimenting left to do in order to fully take advantage of the deep learning techniques at our disposal. Considering this, in section 5.2 we will briefly expose the main issues to be solved, as well as some possible solutions and future work to be done in this area.

5.2 Future Work

We will start by presenting possible solutions to the general issues that are still unsolved by our methods and also possible improvements to what we developed:

- Implement the proposed convolutional neural network (section 3.3.1) to improve the detection of non-informative. In order to do that, it would be necessary to create a new dataset, manually dividing several images into informative and non-informative.
- With the smart video compiler used to compile the endoscopic capsule videos in *Given Imaging* software, the frame rate is drastically reduced (to around 0.7 *fps*). With that in mind, it would be truly important to start cooperating with *Given Imaging* in order to be able to decrypt and read the raw uncompiled videos. In case that is not possible, a method to detect when there is not enough superimposition between two frames might need to be developed.

When it comes specifically to the displacement estimation framework:

- In order to further validate the displacement results obtained with our method, it would be important to create a few more fully labeled videos. This labeling could be done by manually associating the displacement percentages obtained with the *Given Imaging* hardware to each individual image frame throughout the small intestine.
- Alternatively to the displacement method we proposed, a recurrent neural network could be trained to obtain the per frame percentage displacement in an end-to-end manner (e.g. LSTM network). In order to accomplish this task we would need to create a considerably sized dataset, which could also be manually created by attaching the percentages obtained with the *Given Imaging* hardware to each individual image frame in several endoscopic videos.

Finally, for the second, 3D pose and depth estimation framework:

- Once a method to identify non-informative frames is defined, we need to determine a value to replace the discarded measures with. We suggest the use of the average capsule per frame displacement (calculated in equation 4.6) with the last valid predicted capsule direction.
- In order to obtain better results from the training process, it is critical to eliminate most of the non-informative frames. For this, a new dataset generation process needs to be developed. Each time a non-informative frame is detected and discarded, the video needs to be divided in order to maintain only consecutive frames. This new process will need to ensure that the newly created excerpt is at least as long as the defined snippet size (section 4.2.1).
- The results would probably improve if we multiplied the photometric loss of the pixels we want to dismiss by zero. This would be important so that the optimizer does not compare pixels in the black borders with informative pixels.

- In order to further evaluate our method, it would be crucial to have access to an endoscopic video with an attached hardware-based accurate 3D localization. Ideally, this would be done in a real environment, but, due to obvious limitations, the possibility of using a small intestine model with similar properties is definitely not rejected.
- Implement a 3D reconstruction from depth maps method.

References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- Alaa E Abdel-Hakim and Aly A Farag. Csift: A sift descriptor with color invariant characteristics. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1978–1983. IEEE, 2006.
- Sohailah Alyammahi, Ehab Salahat, Hani Saleh, and Andrzej Sluzek. A hardware accelerator for real-time extraction of the linear-time msr algorithm. In *Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE*, pages 000065–000069. IEEE, 2015.
- Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.
- Simon Baker, Ankur Datta, and Takeo Kanade. Parameterizing homographies. *Technical Report CMU-RI-TR-06-11*, 2006.
- Guanqun Bao, Liang Mi, and Yishuang Geng. A computer vision based speed estimation technique for localizing the wireless capsule endoscope inside small intestine. 2014.
- Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.
- Pascal Bertolino. Sensarea, a general public video editing application. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3429–3431. IEEE, 2014.
- Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.
- Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.
- Matthew Brown, David G Lowe, et al. Recognising panoramas. In *ICCV*, volume 3, page 1218, 2003.
- Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 510–517. IEEE, 2005.
- Federico Carpi, Nathan Kastelein, Michael Talcott, and Carlo Pappone. Magnetically controllable gastrointestinal steering of video capsules. *IEEE Transactions on Biomedical Engineering*, 58(2):231–234, 2011.

- Warren Cheung and Ghassan Hamarneh. N-sift: N-dimensional scale invariant feature transform for matching medical images. In *Biomedical Imaging: From Nano to Macro, 2007. ISBI 2007. 4th IEEE International Symposium on*, pages 720–723. IEEE, 2007.
- Hélder Filipe Pinto de Oliveira. *An affordable and practical 3d solution for the aesthetic evaluation of breast cancer conservative treatment*. PhD thesis, Universidade do Porto (Portugal), 2013.
- Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.
- Nilanjan Dey, Amira S Ashour, Fuqian Shi, and R Simon Sherratt. Wireless capsule gastrointestinal endoscopy: Direction-of-arrival estimation based localization survey. *IEEE reviews in biomedical engineering*, 10:2–11, 2017.
- Michael Donoser and Horst Bischof. 3d segmentation by maximally stable volumes (msvs). In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 63–66. IEEE, 2006.
- Elan Dubrofsky. Homography estimation. *Diplomová práce. Vancouver: Univerzita Britské Kolumbie*, 2009.
- Yichen Fan, Max Q-H Meng, and Baopu Li. 3d reconstruction of wireless capsule endoscopy images. In *Engineering In Medicine And Biology Society (Embc), 2010 Annual International Conference Of The Ieee*, pages 5149–5152. IEEE, 2010.
- Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- Michael Fluckiger and Bradley J Nelson. Ultrasound emitter localization in heterogeneous media. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 2867–2870. IEEE, 2007.
- John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world’s imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016.
- Per-Erik Forssén. Maximally stable colour regions for recognition and matching. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010.
- Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.
- Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3):335–360, 2011.
- Zoubin Ghahramani. Unsupervised learning. In *Advanced lectures on machine learning*, pages 72–112. Springer, 2004.

- Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.
- Mahesh K Goenka, Shounak Majumder, and Usha Goenka. Capsule endoscopy: Present status and future expectation. *World Journal of Gastroenterology: WJG*, 20(29):10024, 2014.
- Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Oscar G Grasa, J Civera, A Guemes, V Munoz, and JMM Montiel. Ekf monocular slam 3d modeling, measuring and augmented reality from endoscope image sequences. In *Medical image computing and computer-assisted intervention (MICCAI)*, volume 2, 2009.
- Oscar G Grasa, Ernesto Bernal, Santiago Casado, Ismael Gil, and JMM Montiel. Visual slam for handheld monocular endoscope. *IEEE transactions on medical imaging*, 33(1):135–146, 2014.
- Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.
- Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- M Hassaballah, Aly Amin Abdelmgeid, and Hammam A Alshazly. Image features detection, description and matching. In *Image Feature Detectors and Descriptors*, pages 11–45. Springer, 2016.
- EM Haworth, CJ Hodson, CRB Joyce, EM Pringle, G Solimano, and WF Young. Radiological measurement of small bowel calibre in normal subjects according to age. *Clinical radiology*, 18(4):417–421, 1967.
- Xiaoqi He, Zizhao Zheng, and Chao Hu. Magnetic localization and orientation of the capsule endoscope based on a random complex algorithm. *Medical devices (Auckland, NZ)*, 8:175, 2015.
- Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- Chao Hu, Max Q-H Meng, and Mrinal Mandal. Efficient magnetic localization and orientation technique for capsule endoscopy. *International Journal of Information Acquisition*, 2(01):23–36, 2005.
- Dimitris K Iakovidis and Anastasios Koulaouzidis. Software for enhanced video capsule endoscopy: challenges for essential progress. *Nature Reviews Gastroenterology & Hepatology*, 12(3):172–186, 2015.
- Dimitris K Iakovidis, Evaggelos Spyrou, Dimitris Diamantis, and Ilias Tsiompanidis. Capsule endoscope localization based on visual features. In *Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on*, pages 1–4. IEEE, 2013.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

- Francesco Isgro and Emanuele Trucco. Projective rectification without epipolar geometry. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, pages 94–99. IEEE, 1999.
- Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.
- Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004.
- Henrik Keller, Aleksandar Juloski, Hironao Kawano, Mario Bechtold, Atsushi Kimura, Hironobu Takizawa, and Rainer Kuth. Method for navigation and control of a magnetically guided capsule endoscope in the human stomach. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, pages 859–865. IEEE, 2012.
- Dang Trung Kien. A review of 3D reconstruction from video sequences. *University of Amsterdam ISIS Technical Report Series*, 2005.
- Kang Kim, Laura A Johnson, Congxian Jia, Joel C Joyce, Sujal Rangwalla, Peter DR Higgins, and Jonathan M Rubin. Noninvasive ultrasound elasticity imaging (uei) of crohn’s disease: animal model. *Ultrasound in Medicine and Biology*, 34(6):902–912, 2008.
- Reinhard Koch, Marc Pollefeys, and Luc Van Gool. Realistic surface reconstruction of 3d scenes from uncalibrated image sequences. *The Journal of Visualization and Computer Animation*, 11(3):115–127, 2000.
- Axel Krieger, Iulian I Iordachita, Peter Guion, Anurag K Singh, Aradhana Kaushal, Cynthia Ménard, Peter A Pinto, Kevin Camphausen, Gabor Fichtinger, and Louis L Whitcomb. An mri-compatible robotic system with hybrid tracking for mri-guided prostate intervention. *IEEE Transactions on Biomedical Engineering*, 58(11):3049–3060, 2011.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015. ISSN 0028-0836. doi: 10.1038/nature14539. URL <http://www.nature.com/doifinder/10.1038/nature14539>.
- Man Hee Lee and In Kyu Park. Performance evaluation of local descriptors for affine invariant region detector. In *Asian Conference on Computer Vision*, pages 630–643. Springer, 2014.
- Maxime Lhuillier and Long Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE transactions on pattern analysis and machine intelligence*, 27(3):418–433, 2005.
- Jianguo Li, Eric Li, Yurong Chen, Lin Xu, and Yimin Zhang. Bundled depth-map merging for multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2769–2776. IEEE, 2010.

- Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. *arXiv preprint arXiv:1709.06841*, 2017.
- Bingxiong Lin, Adrian Johnson, Xiaoning Qian, Jaime Sanchez, and Yu Sun. Simultaneous tracking, 3d reconstruction and deforming point detection for stereoscope guided surgery. In *Augmented Reality Environments for Medical Imaging and Computer-Assisted Interventions*, pages 35–44. Springer, 2013.
- Tony Lindeberg. Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, 52(1):3–36, 2015.
- Li Ling, Ian S Burrent, and Eva Cheng. A dense 3d reconstruction approach from uncalibrated video sequences. In *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, pages 587–592. IEEE, 2012.
- Qiang Liu, Ruihao Li, Huosheng Hu, and Dongbing Gu. Extracting semantic information from visual data: A survey. *Robotics*, 5(1):8, 2016.
- Shuai Liu, Lingli Zhao, Junsheng Li, and Haicheng Xu. The research of 3d reconstruction from uncalibrated image sequences combined with 3d models. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 2, pages 1117–1119. IEEE, 2008.
- Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
- Charles Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, pages 125–131. IEEE, 1999.
- David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.
- Nader Mahmoud, Iñigo Cirauqui, Alexandre Hostettler, Christophe Doignon, Luc Soler, Jacques Marescaux, and JMM Montiel. Orbslam-based endoscope tracking and 3d reconstruction. In *International Workshop on Computer-Assisted and Robotic Endoscopy*, pages 72–83. Springer, 2016.
- Arthur W Mahoney, Samuel E Wright, and Jake J Abbott. Managing the attractive magnetic force between an untethered magnetically actuated tool and a rotating permanent magnet. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5366–5371. IEEE, 2013.
- Neil Marya, Andrew Karellas, Anne Foley, Abhijit Roychowdhury, and David Cave. Computerized 3-dimensional localization of a video capsule in the abdominal cavity: validation by digital radiography. *Gastrointestinal endoscopy*, 79(4):669–674, 2014.
- Jiri Matas, Ondrej Chum, Martin Urban, and Tomáš Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

- Haris Mateen, Rubel Basar, Afaz Uddin Ahmed, and Mohd Yazed Ahmad. Localization of wireless capsule endoscope: A systematic review. *IEEE Sensors Journal*, 17(5):1197–1206, 2017.
- Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.
- Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.
- Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–1630, 2005.
- Ondrej Miksik and Krystian Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2681–2684. IEEE, 2012.
- Peter Mountney, Danail Stoyanov, Andrew Davison, and Guang-Zhong Yang. Simultaneous stereoscope localization and soft-tissue mapping for minimal invasive surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 347–354. Springer, 2006.
- Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.
- Fredy Munoz, Gursel Alici, and Weihua Li. A review of drug delivery systems for capsule endoscopy. *Advanced drug delivery reviews*, 71:77–85, 2014.
- Tetsuya Nakamura and Akira Terano. Capsule endoscopy: past, present, and future. *Journal of gastroenterology*, 43(2):93–99, 2008.
- Ty Nguyen, Steven W Chen, Shreyas S Shivakumar, Camillo J Taylor, and Vijay Kumar. Unsupervised deep homography: A fast and robust homography estimation model. *arXiv preprint arXiv:1709.03966*, 2017.
- David Nistér and Henrik Stewénus. Linear time maximally stable extremal regions. *Computer Vision–ECCV 2008*, pages 183–196, 2008.
- Søren I Olsen. Epipolar line estimation. In *European Conference on Computer Vision*, pages 307–311. Springer, 1992.
- Zhigeng Pan, Xianyong Fang, Jiaoying Shi, and Dan Xu. Easy tour: a new image-based virtual tour system. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 467–471. ACM, 2004.
- Andrew J Petruska and Jake J Abbott. An omnidirectional electromagnet for remote manipulation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 822–827. IEEE, 2013.
- Duc Minh Pham and Syed Mahfuzul Aziz. A real-time localization system for an endoscopic capsule using magnetic sensors. *Sensors*, 14(11):20910–20929, 2014.

- M Rafiei and M Saadatseresht. Automatic 3d mapping using multiple uncalibrated close range images. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, (3):327–332, 2013.
- Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE, 2005.
- Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.
- JM Rubin, H Xie, and K Kim. Sonographic elasticity imaging of acute and chronic deep venous thrombosis in humans. *Journal of Vascular Surgery*, 45(5):1087, 2007.
- Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.
- Ehab Salahat and Murad Qasaimeh. Recent advances in features extraction and description algorithms: A comprehensive survey. In *Industrial Technology (ICIT), 2017 IEEE International Conference on*, pages 1059–1063. IEEE, 2017.
- Ehab Salahat, Hani Saleh, Safa Salahat, Andrzej Sluzek, Mahmoud Al-Qutayri, B Mohammed, and Mohammad Ismail. Extended msr detection. In *The IEEE International Symposium on Industrial Electronics*, pages 3–5, 2015a.
- Ehab Salahat, Hani Saleh, Andrzej Sluzek, Mahmoud Al-Qutayri, Baker Mohammad, and Mohammad Ismail. A maximally stable extremal regions system-on-chip for real-time visual surveillance. In *Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE*, pages 002812–002815. IEEE, 2015b.
- Ehab Najeh Salahat, Hani Hasan Mustafa Saleh, Andrzej Stefan Sluzek, Mahmoud Al-Qutayri, Baker Mohammad, and Mohammed Ismail Elnaggar. Architecture and method for real-time parallel detection and extraction of maximally stable extremal regions (msers), April 12 2016. US Patent 9311555B2.
- Ehab Najeh Salahat, Hani Hasan Mustafa Saleh, Safa Najeh Salahat, Andrzej Stefan Sluzek, Mahmoud Al-Qutayri, Baker Mohammad, and Mohammed Ismail Elnaggar. Object detection and tracking using depth data, May 2 2017. US Patent 9,639,951.
- Radim Šára. Finding the largest unambiguous component of stereo matching. In *European Conference on Computer Vision*, pages 900–914. Springer, 2002.
- Mohit Shridhar and Kai-Yuan Neo. Monocular slam for real-time applications on mobile platforms. 2015.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

- Metin Sitti, Hakan Ceylan, Wenqi Hu, Joshua Giltinan, Mehmet Turan, Sehyuk Yim, and Eric Diller. Biomedical applications of untethered mobile milli/microrobots. *Proceedings of the IEEE*, 103(2):205–224, 2015.
- Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.
- Evaggelos Spyrou and Dimitris K Iakovidis. Homography-based orientation estimation for capsule endoscope tracking. In *Imaging Systems and Techniques (IST), 2012 IEEE International Conference on*, pages 101–105. IEEE, 2012. ISBN 1457717751.
- Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.
- Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- Cheng-Yuan Tang, Yi-Leh Wu, Pei-Ching Hu, Hsien-Chang Lin, and Wen-Chao Chen. Self-calibration for metric 3d reconstruction using homography. In *MVA*, pages 86–89, 2007.
- Trung Duc Than, Gursel Alici, Hao Zhou, and Weihua Li. A review of localization systems for robotic endoscopic capsules. *IEEE transactions on biomedical engineering*, 59(9):2387–2399, 2012.
- Mehmet Turan, Abdullah Abdullah, Redhwan Jamiruddin, Helder Araujo, Ender Konukoglu, and Metin Sitti. Six degree-of-freedom localization of endoscopic capsule robots using recurrent neural networks embedded into a convolutional neural network. *arXiv preprint arXiv:1705.06196*, 2017a.
- Mehmet Turan, Yasin Almalioglu, Ender Konukoglu, and Metin Sitti. A deep learning based 6 degree-of-freedom localization method for endoscopic capsule robots. *CoRR*, abs/1705.05435, 2017b. URL <http://arxiv.org/abs/1705.05435>.
- Mehmet Turan, Yasin Almalioglu, Helder Araujo, Ender Konukoglu, and Metin Sitti. Deep endo: A recurrent convolutional neural network (rcnn) based visual odometry approach for endoscopic capsule robots. *Neurocomputing*, 275:1861–1870, 2018.
- Tinne Tuytelaars, Krystian Mikolajczyk, et al. Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280, 2008.
- Ilknur Umay. Adaptive wireless biomedical capsule localization and tracking. Master’s thesis, University of Waterloo, 2015.
- Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 5, 2017.
- Cedric Van de Bruaene, Danny De Looze, and Pieter Hindryckx. Small bowel capsule endoscopy: Where are we after almost 15 years of use? *World journal of gastrointestinal endoscopy*, 7(1):13, 2015.
- Andrea Vedaldi. An implementation of multi-dimensional maximally stable extremal regions. *Feb*, 7:1–7, 2007.

- Etienne Vincent and Robert Laganier. Matching feature points in stereo pairs: A comparative study of some matching strategies. *Machine Graphics and Vision*, 10(3):237–260, 2001.
- George Vogiatzis, Carlos Hernández Esteban, Philip HS Torr, and Roberto Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, 2007.
- Haibing Wu and Xiaodong Gu. Max-pooling dropout for regularization of convolutional neural networks. In *International Conference on Neural Information Processing*, pages 46–54. Springer, 2015.
- Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.
- Shuqiang Yang and Biao Li. Outliers elimination based ransac for fundamental matrix estimation. In *Virtual Reality and Visualization (ICVRV), 2013 International Conference on*, pages 321–324. IEEE, 2013.
- Sehyuk Yim and Metin Sitti. 3-d localization method for a magnetically actuated soft capsule endoscope and its applications. *IEEE Transactions on Robotics*, 29(5):1139–1151, 2013.
- Sehyuk Yim, Evin Gultepe, David H Gracias, and Metin Sitti. Biopsy using a magnetic capsule endoscope carrying, releasing, and retrieving untethered microgrippers. *IEEE Transactions on Biomedical Engineering*, 61(2):513–521, 2014.
- Zhongfei Zhang and Allen R Hanson. 3d reconstruction based on homography mapping. *Proc. ARPA96*, pages 1007–1012, 1996.
- Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.