Bachelor Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Circuit Theory

Histological image registration using optical flow estimation and deep learning

Vojtěch Brejtr

Supervisor: prof. Dr. Ing. Jan Kybic
Field of study: Medical electronics and bioinformatics
May 2022

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Brejtr Vojtěch**	Personal ID number:	**491929**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Circuit Theory**		
Study program:	**Medical Electronics and Bioinformatics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Histological image registration using optical flow estimation and deep learning

Bachelor’s thesis title in Czech:

Registrace histologických snímků za pomoci optického toku a hlubokého učení

Guidelines:

The goals of this work are to evaluate the performance of existing deep-learning-based optical flow estimation algorithms for registration of histological images from the ANHIR dataset (https://anhir.grand-challenge.org/), comparing their results with known results of other state-of-the-art registration algorithms, and adapting the optical-flow algorithm to improve their performance on this data.
Instructions:
1. Get acquainted with the ANHIR dataset and at least three existing deep-learning-based optical flow estimation methods.
2. Apply the optical flow methods on the ANHIR dataset, visualize and evaluate the results using the ANHIR challenge methodology and compare them with reported results of other methods.
3. Analyze the weak points of the optical flow methods on this dataset and suggest possible improvements. Consider changes in the training procedure (both supervised and unsupervised), the similarity function, or the multiresolution strategy.
4. Implement and experimentally evaluate the suggested changes.
5. [Optional] Evaluate the generalization ability of the developed method by testing on other datasets.

Bibliography / sources:

[1] SUN, Deqing, et al. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. p. 8934-8943.
[2] TEED, Zachary, DENG, Jia. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In: European Conference on Computer Vision. Springer, Cham, 2020. p. 402-419.
[3] JONSCHKOWSKI, Rico, et al. What Matters in Unsupervised Optical Flow. In: European Conference on Computer Vision. Springer, Cham, 2020. p. 557-572.
[4] DOSOVITSKIY, Alexey, et al. FlowNet: Learning Optical Flow with Convolutional Networks. In: Proceedings of the IEEE International Conference on Computer Vision. 2015. p. 2758-2766.
[5] BOROVEC J., Kybic J., Arganda-Carreras I., Sorokin D.V., Bueno G., Khvostikov A.V., Bakas S., Chang E. I., Heldmann S., Kartasalo K., Latonen L., Lotz J., Noga M., Pati S., Punithakumar K., Ruusuvuori P., Skalski A., Tahmasebi N., Valkonen M., Venet L., et al. "ANHIR: Automatic Non-rigid Histological Image Registration Challenge." IEEE Transactions on Medical Imaging, no. 10, pp. 3042-3052, October 2020.

Name and workplace of bachelor’s thesis supervisor:

prof. Dr. Ing. Jan Kybic Biomedical imaging algorithms FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **21.02.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **19.02.2024**

prof. Dr. Ing. Jan Kybic	doc. Ing. Radoslav Bortel, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I would like to thank my supervisor, prof. Dr. Ing. Jan Kybic, for the guidance and help during my bachelor thesis.

I would also like to thank my friends who got me through the last year.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 18. May 2022

Abstract

Registration of medical images is an important topic that has seen much research in the last several years, thanks to the advancements in deep neural network architectures. Optical flow estimation methods are generally used for finding pixel-wise motion in image sequences in scenes with constant brightness and homogeneous local movement.

The application of optical flow for the purpose of registration of the histological tissue samples is relatively unexplored. We will use the ANHIR dataset, which contains such samples. This thesis will analyze the main weak points of the optical flow methods on medical images and develop a method capable of correctly registering them. We also propose a training procedure and training dataset generation for this purpose.

The developed method uses the GMA or RAFT architectures, which we trained on our dataset, combining supervised and unsupervised loss functions. The method utilizes a two-part process registering global and local movement separately.

The proposed method is compared to methods submitted to the ANHIR challenge. Our method outperforms all baseline methods provided by the organizer while being competitive with the other participants.

Optical flow is shown to be able to register histological images both locally and globally.

Keywords: machine learning, neural network, deep learning, image registration, optical flow, biomedical imaging, histological images

Supervisor: prof. Dr. Ing. Jan Kybic

Abstrakt

Registrace lékařských snímků je důležité téma, které se v posledních několika letech dočkalo velkého množství výzkumu, díky pokrokům v architekturách hlubokých neuronových sítí. Metody odhadující optický tok jsou obecně používány k nalezení pohybu každého pixelu v sekvenci obrazů s konstantním jasem a lokálně stejnorodým pohybem.

Použití optického toku na histologických vzorcích tkáně je relativně neprozkoumané. Použijeme dataset ANHIR, který tyto vzorky obsahuje. V této práci zanalyzujeme hlavní slabiny metod založených na optickém toku, při použití na lékařských snímcích a vyvineme metodu, která je schopná jejich registrace. Také navrhneme postup trénování a generace trénovacích dat, kterých k tomuto použijeme.

Vyvinutá metoda, využívající architektury GMA nebo RAFT, byla natrénována na našem datasetu, za pomoci učení s učitelem i bez učitele. Tato metoda je založena na dvoudílném procesu, kde dojde k registraci globálního a lokálního pohybu samostatně.

Navrhnutá metoda je následně porovnána s metodxamy ostatních účastníků soutěže ANHIR. Překonala všechny metody poskytnuté organizátory soutěže a je konkurence schopná v porovnání s ostatními účastníky.

Optický tok, se ukázal být schopný registrace histologických snímků jak lokálně, tak globálně.

Klíčová slova: strojové učení, neuronové sítě, hluboké učení, registrace obrazu, optický tok, biomedicínské zobrazování, histologické snímky

Překlad názvu: Registrace histologických snímků za pomoci optického toku a hlubokého učení

Contents

Figures

Tables

Chapter 1

Introduction

The apparent optical flow is used for motion estimation on many different image types and sequences. However, its use for the registration of histological images is relatively unexplored. Therefore, we will develop a method utilizing the optical flow for both the global and local registration of such images.

The analysis of histological tissue is currently the only definitive method for the confirmation of the presence or absence of some diseases, grading of the disease, and quantitative measurements of the disease progression [1]. Over the past several decades, due to the ever-increasing digitization and development of whole slide digital scanners, histopathological slides are becoming more accessible. Tasks, which in the past must have been carried out by medical personnel, are now subject to automation, with the help of computer-assisted diagnosis (CAD), e.g., image alignment for further processing (segmentation), and are becoming a standard part of the workflow in many applications [1].

Development of more accurate methods was accelerated by the rise in popularity of convolutional neural networks due to their immense ability to solve many of the problems facing the medical image analysis community [2].

1.1 Motivation

The registration of histological images is needed for many different reasons. Some of which, amongst many others, are

- Creation of a 3D object from scanned thin 2D slices, by overlaying them on top of one another [3].
- Distortion compensation, after inaccurate initial alignment [4].
- Connection of small patches together to form a high-resolution image [4].
- Fusion of information from different modalities or after different stains have been used on them [5].
- Combination of gene expression maps from multiple specimens [6]

Thanks to machine learning, specifically neural networks, registration of medical images has become more approachable, and the overall process achieves way higher performance than non-machine learning methods. The optical flow estimation deep learning networks have shown to be highly accurate when computing the per-pixel motion in the image.

One of the most critical issues faced in medical image analysis today is the scarcity of high-quality annotated data appropriate for the given task. Many different collections of such data were created over the years, but primarily focusing on image classification or segmentation rather than registration. Their value cannot be overstated as they lower the barrier of entry tremendously.

Challenges are organized to benchmark methods developed for the given dataset. Participants in such challenges are tasked with creating an ML method that is later evaluated. Such challenges usually end with a workshop and discussion between all participating sides in hopes of further improvement in the given field. Some of the challenges with the image registration are the

- **RIRE** - Retrospective Intermodality Brain Image Registration is a project focusing on brain images from the CT, MRI, and PET scans [7].
- **EMPIRE10**[1] - EMPIRE10 challenge focused on CT scans of the lungs [8].
- **CuRIOUS 2019**[2] - Correction of Brain shift with Intra-Operative Ultrasound focuses on the registration of pre-operative MRI to iUS before brain tumor resection and the registration of iUS after-before brain tumor resection.

and the ANHIR Challenge[3] [9], which we will use as the benchmark for our experiments.

1.1.1 ANHIR Challenge

The Automatic Non-rigid Histological Image Registration Challenge (ANHIR) aimed to evaluate the accuracy, robustness, and speed of non-rigid registration methods on microscopy histology images [9]. These images have been stained by a collection of several histological dyes (see section 2.3). Images are from eight distinct groups (e.g., lung lesions, human breast tissue, or human kidney tissue), each of which is annotated with landmarks, which serve as a basis for the method evaluation (see section 3.5). Part of the dataset was previously made public as a part of a research paper [10].

Images were provided by various organizations, such as CIMA of the University of Navarra, Institute of Pathology of University Hospital Aachen, Masaryk Memorial Cancer Institute Brno, Masaryk University Brno, Department of Pathology of Lomonosov Moscow State University, Grupo VISILAB

[1]Available at `https://empire10.grand-challenge.org/`
[2]Available at `https://curious2019.grand-challenge.org/`
[3]Available at `https://anhir.grand-challenge.org/`

of UCLM. Images were obtained and prepared thanks to the AIDPATH European project [9].

1.2 Goals and structure of this thesis

The main goals of this thesis are as follows.

- Development of a medical image registration method based on optical flow estimation. Optical flow is generally not used in this context but rather in longer image sequences, like autonomous driving, image tracking, video compression, and movement detection, amongst many other applications [11].
- Establish the effectiveness of the optical flow methods on medical images.
- Propose possible improvements to both the method and the training procedure.

In the chapters that follow, a method based on optical flow estimation is created and tested on the ANHIR [9] dataset. Its results are evaluated and compared to the other participants of the ANHIR challenge [9].

This work also contains additional information, which should allow the reader to better understand the medical background, other registration methods, and the problem of the optical flow estimation.

Chapter 2
Medical background

Histology, also known as microscopic anatomy, studies the microscopic anatomy of diseased tissue samples through staining, sectioning, and examination under a microscope. Its importance lies in the diagnosis of a wide variety of diseases, for example, cancer, in order to aid future treatment [12].

Histological staining is a series of processes in the preparation of the tissue by staining it with special chemicals in order to make different features more pronounced or visible for the microscope study [13]. This process has five distinct stages: fixation, processing, embedding, sectioning, and staining.

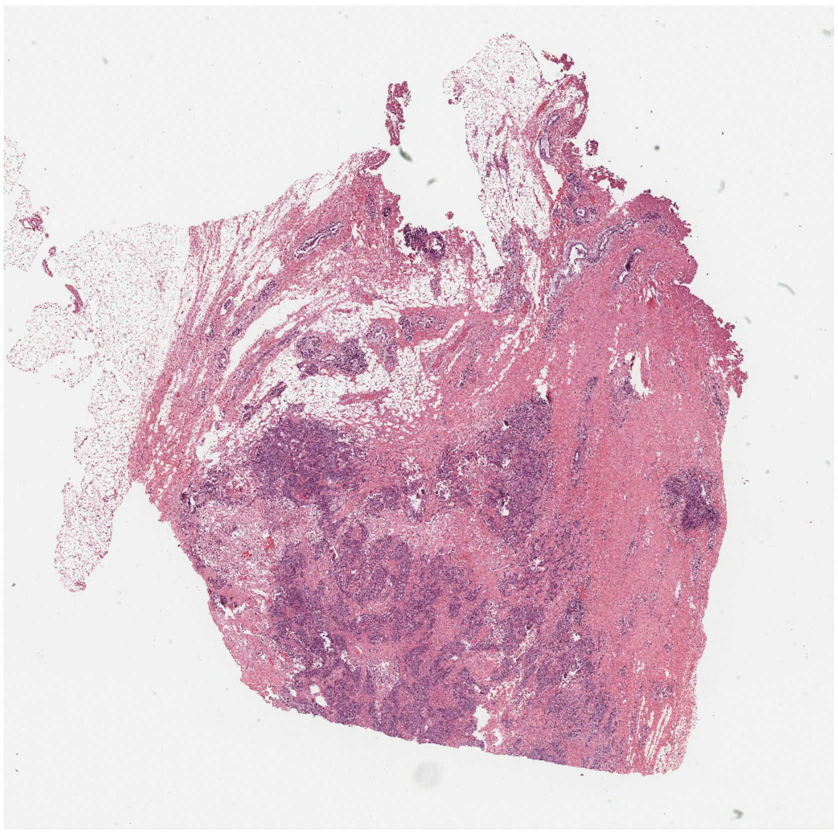


Figure 2.1: Human breast tissue stained with H&E

2.1 Sample preparation

2.1.1 Fixation

Fixation provides a way to prevent the natural tissue and cellular structure from degrading. For the microscopy using a light microscope, neutral buffered formalin (NBF) is used. Fixatives irreversibly cross-link proteins. Unfortunately, in the process, DNA and mRNA, amongst many other protein structures, undergo denaturalization and, as such, may lead to incorrect results [13]. Nonetheless, the tissue retains its chemical composition and hardens the cells.

They also change the penetrative properties of the tissue, possibly enhancing the effectiveness of stains. Aside from NBF, other commonly used fixatives are paraffin-formalin or Bouin fixative [14].

2.1.2 Processing

Processing consists of several different steps, one of them being dehydration. After the removal of the water from the sample, the tissue is solidified and cut into thin slices (thinner for electron microscopy, somewhat thick for light microscopy) [15]. Once more, the process is repeated through a hydrophobic clearing substance such as xylene to remove unwanted chemicals like alcohol or paraffin wax.

2.1.3 Embedding

This part of the overall process is to make the extraction of the cellular structures easier and preserve the morphology of the tissue. Paraffin wax, or in more complex structures, plastic resin, is used. However, the same problem presents itself as with other fixatives, being the possibility of degrading the cellular structure [13].

2.1.4 Sectioning

Sectioning refers to the preparation of thin ribbon-like slices, which are then used in the microscope [13].

2.1.5 Staining

As stated above, staining is used to highlight important features of the tissue. One of the most frequently used stains is hematoxylin, which stains the nuclei with a bluish color, commonly used together with eosin, which stains the nuclei with a pinkish color, giving tissues a familiar look as shown in figure 2.1. There is, however, a wide variety of different staining techniques for specific cells and tasks [13]. Staining is often used for the diagnosis of tumors in which a dye color is applied to the front and the back of the sample to locate tumorous or diseased cells [16].

2.2 Histological tissue

This section will describe several different types of histological images from different tissue samples, stains used on them, and diseases diagnosed with the help of said images. All image groups of this section are represented in the ANHIR dataset [9].

2.2.1 Lung lesion

Lung lesions, sometimes referred to as pulmonary nodules, are defined as an approximately rounded opacity [17], which is more or less well-defined, measuring up to 3 cm in diameter. Lesions whose diameter is above 3 cm are considered lung masses and indicate lung cancer. The prevalence of such masses is around 40%, but varies widely depending on different things such as age, medical history, or smoking [18]. They can be categorized into four different types,

- **Benign tumors**, such as hamartomas, which is a local malformation of cells caused by an overgrowth of multiple aberrant cells. They account for about 75% of all benign lung tumors and around 7% of all solitary lung nodules. [19]
- **Infections**, such as tuberculosis, which accounts for around 7% of all deaths in developing countries, making it the single highest cause of death from a single source of infection among adults. The symptoms of tuberculosis are chronic cough, pain in the chest, hemoptysis, weakness or fatigue, and many others [20]. Other sources of infection can be fungal in origins, such as histoplasmosis or coccidioidomycosis [21].
- **Inflammation**, such as rheumatoid arthritis or sarcoidosis [22].
- **Malignant tumors**, including lung cancer and cancer that has spread to the lung from other parts of the body. Lung cancer is a leading cause of cancer deaths worldwide, with 1.8 million new cases and 1.6 million deaths annually. Its survival rate for a 5-year period is 18%, compared to breast, colon, or prostate cancer, which all have a rate of 90, 65, and nearly 100 %, respectively. The majority of lung cancer cases are diagnosed at the advanced stage [23].

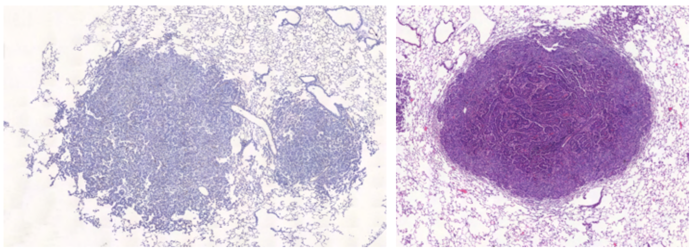


Figure 2.2: Lung lesions from the ANHIR dataset [9]

2.2.2 Lung lobes

The lungs are the primary respiratory system in most animals. Human lungs are comprised of lung lobes. The right lung has three lobes (superior, middle, and inferior), and the left has only two lobes (superior and inferior) due to the presence of the heart. However, in rodents, lung lobes are distributed differently, with the right lung having an additional post-caval lobe and the left one being comprised of a single lobe [24]. Our images contain the lung lobes of mice.

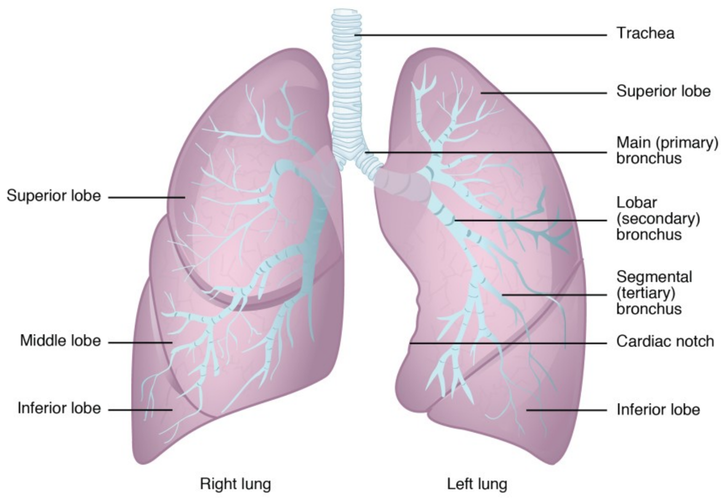


Figure 2.3: Gross anatomy of the human lungs, taken from [25]

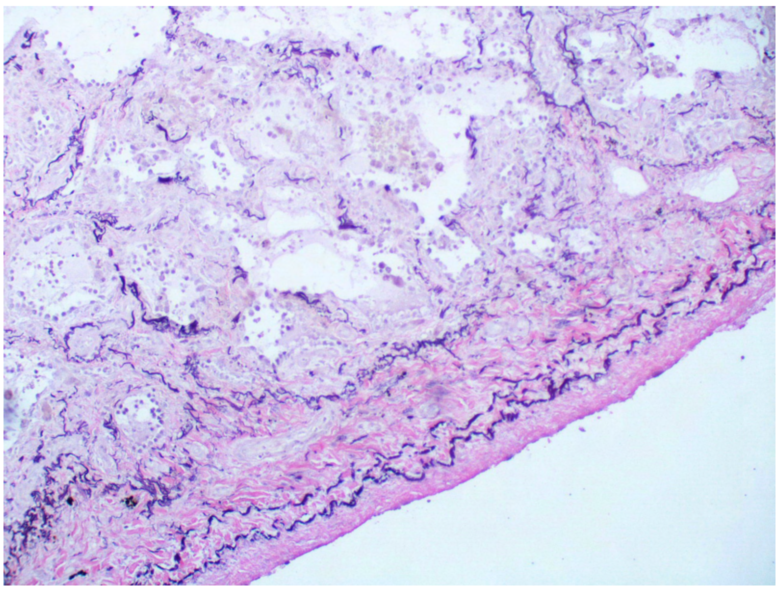


Figure 2.4: Lung tissue, taken from [26]

2.2.3 Breast tissue

Histological slices of human breast tissue are most commonly obtained for breast cancer research. Breast cancer is globally the leading cause of mortality among women. In 2018, a total of 2.1 million women were diagnosed, and around 630 thousand death were recorded [27]. The primary distinction is whether the tumor is limited to the epithelial component of the breast or has spread to the surrounding stroma.

Tumor types can also be separated based on where its located, being either in the ducts or lobes. In the actual histological practice, more general descriptors are used, such as cell type and their number, location of secretion, or their immunohistochemical profile [28]. Around 50 % to 80% of all diagnosed breast cancer cases are so-called invasive ductal carcinoma (IDC), while the rest are classified as invasive lobular carcinoma (ILC). They both have many subtypes, based on their distinctive characteristics [28]. Some of which are

- **Invasive ductal carcinoma no specific type (IDC-NST)** is the most prevalent type of invasive breast carcinomas, constituting about 40% to 75% of all cases. It has a wide variety of morphological variations and clinical behaviors. Areas with dead and calcified tissue are detectable in more than 50% of the cases [28].

- **Invasive lobular carcinoma** is the second most common breast carcinoma with around 5% to 15% of all cases. It generally affects women of advanced age. It is characterized by small tumor cells with atypia, uniformly distributed in a concentric pattern [29].

- **Medullary carcinoma** is responsible for 5% of all cases. The cause of this cancer type is a mutation of the BRCA1(Breast cancer gene 1) germline. It is well-circumscribed, composed of large and pleomorphic tumor cells, spindle cell metaplasia, and giant tumor cells [28].

- **Apocrine carcinoma** constitutes around 1% to 4% of all cases, with prominent apocrine differentiation comprising at least 90% of tumor cells. This subtype is of high histological grade with a poor prognosis. It affects women of all ages but is more common in women after menopause. Bizarre tumor cells with multilobulated nuclei are also observable [28].

Nowadays, the cancer classification based on its morphological properties such as nuclear and tubular grade, mitotic index or architecture characteristics, and the pathological parameters like tumor size, lymph node involvement, or metastasis is not good enough to predict the real behavior of breast tumor pathophysiology. Classification based on the molecular patterns is therefore utilized [28]. Four clinically relevant molecular subtypes are Luminal A, Luminal B, enriched HER2, and Triple Negative (see section Histological stains and markers)

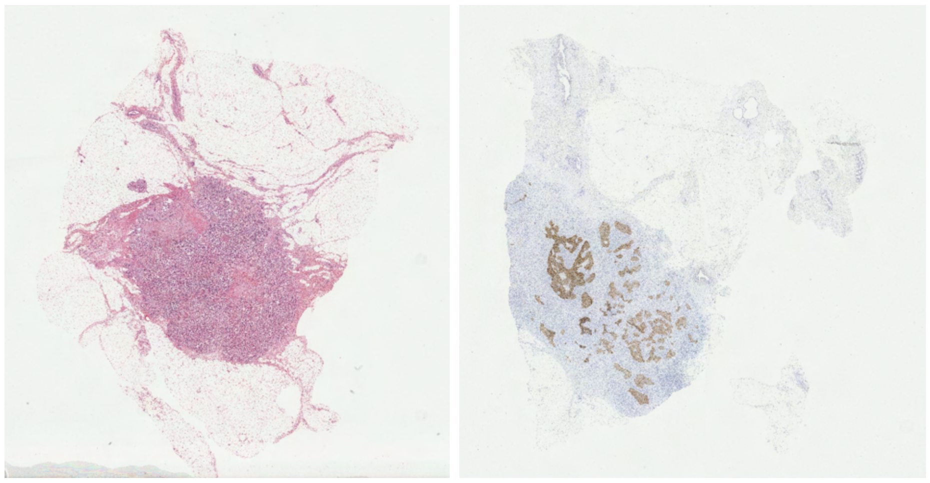


Figure 2.5: Human breast tissue from the ANHIR dataset [9]

2.2.4 Mammary glands

Mammary glands are responsible for the production of breast milk. They are modified sweat glands consisting of alveoli, lined with milk-secreting cuboidal cells and surrounded by myoepithelial cells. Alveoli join together to form lobules, which are in turn drained by a lactiferous duct into the opening in the nipple [30]. Most of the breast carcinoma originates either in the lobes or the ducts (see 2.2.3).

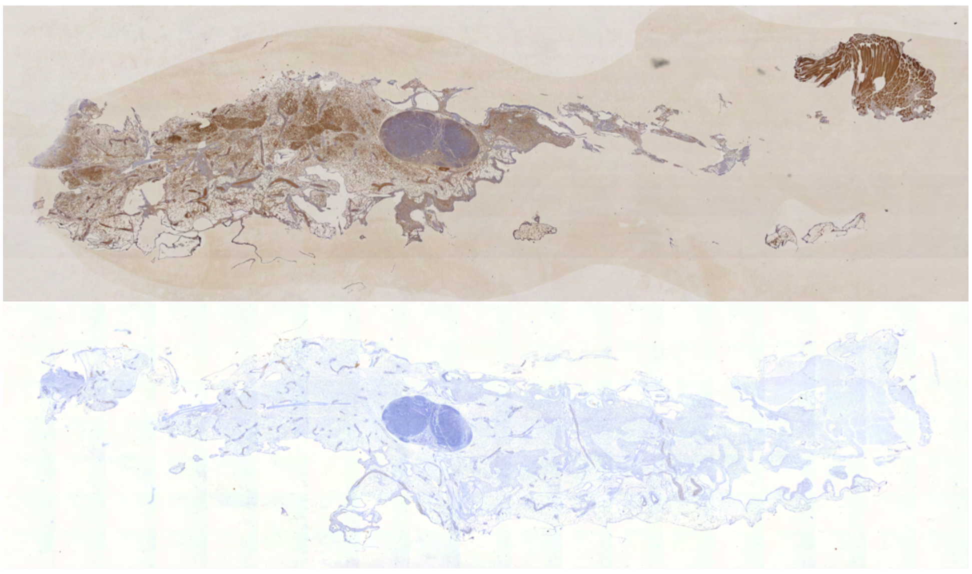


Figure 2.6: Mice mammary glands from the ANHIR dataset [9]

2.2.5 Gastric tissue

As with the breast tissues, gastric samples are commonly collected to further research cancer prevention or therapy. Gastric cancer is the fifth most common cancer and the third most common cause of cancer death globally, with around 1 million new cases and around 780 thousand deaths reported in 2018 [31]. Risk factors for the condition include age, high salt intake, diets low in fruit and vegetables, and Helicobacter pylori infection [32].

From the histological standpoint, early gastric cancers can be classified as

- **Type I** - Protruded
- **Type II** - Superficial (either elevated, flat or depressed)
- **Type III** - Excavated

The advanced stages of the gastric cancer are commonly classified according to Borrmann [32] into four types, being

- **Type I** - polypoid, without elevated borders of sharp margins
- **Type II** - ulcerated with elevated border and sharp margins
- **Type III** - ulcerated with diffuse infiltration at the base
- **Type IV** - diffusely infiltrative thickening of the wall

The five-year survival rate for the patients varies widely between countries (USA - 32%, Japan - 70%). For the advanced stages, specifically Type IV, the survival rate is below 5% [32].

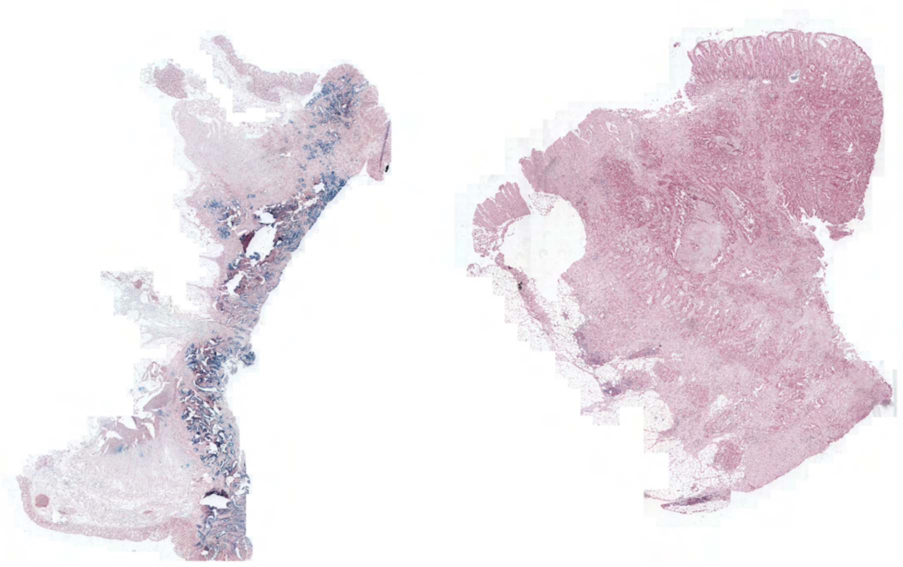


Figure 2.7: Gastric tissue from the ANHIR dataset [9]

2.2.6 Kidney tissue

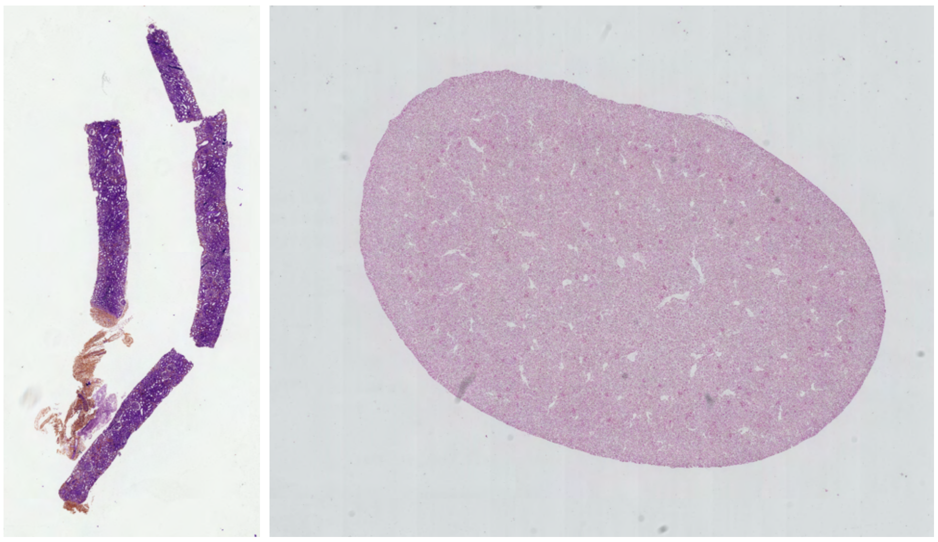
Kidneys are partly responsible for controlling various bodily fluids, acid-base balance, the removal of toxins, and blood pressure regulation, amongst many more functions. The parenchyma of the kidney is divided into two major structures, the outer renal cortex and the inner renal medulla [30].

The main functional units are the nephrons. They are responsible for the formation of the urine by filtering the blood passing through them. They consist of a clump of capillaries called glomerulus and a structure called Bowman's capsule [30].

Once the kidneys cease working, the resulting state is called **chronic kidney disease** (**CKD**). It does not have any symptoms in the early stages and, as such, can only be diagnosed by chance from blood or urine test conducted for other reasons. However, symptoms can include tiredness, swollen ankles, feet or hand, shortness of breath, and blood in the urine at a more advanced stage. It is caused by high blood pressure, diabetes, high cholesterol, infection, or long-term effects of some medications [33].

Some of the kidney diseases that can lead to the CKD are

- **Cystosis**, which is caused by a build-up of a naturally occurring chemical called cystein.
- **Glomerulonephritis**, where the glomeruli are damaged and lose their ability to remove waste and fluid from the blood.
- **IgA Nephroapthy**, caused by a build-up of proteins created by the immune system.



(a) : Glomerulopathies blocks **(b)** : Mice kidney

Figure 2.8: Sections of glomerulopathies blocks and mice kidney from the ANHIR dataset [9]

2.3 Histological stains and markers

Histological stains change the coloration of cells and tissues significantly. This is used to highlight specific structures with the sample [13]. Some of the commonly used stains or markers are

- **Clara cell 10 protein (CC10)**
  The CC10 protein is generated by the Clara cells, which are secretory epithelial cells lining the pulmonary airways. It is primarily used for the detection of pulmonary tumors [34].

- **Hematoxylin and eosin (H&E)**
  H&E is the standard stain used for histological examination of human tissue. The combination of hemotoxylin and eosin is capable of highlighting the fine structures of cells and tissue because most organelles and extracellular matrices are eosinophilic, while the nucleus is rough endoplasmic reticulum and ribosomes are basophilic [35].

- **Antigen KI-67 (Ki67)**
  Ki67 is associated with cell proliferation. It is present in the active phase of the cell cycle but absent in the resting phase. It is therefore capable of serving as a proliferation marker [36].

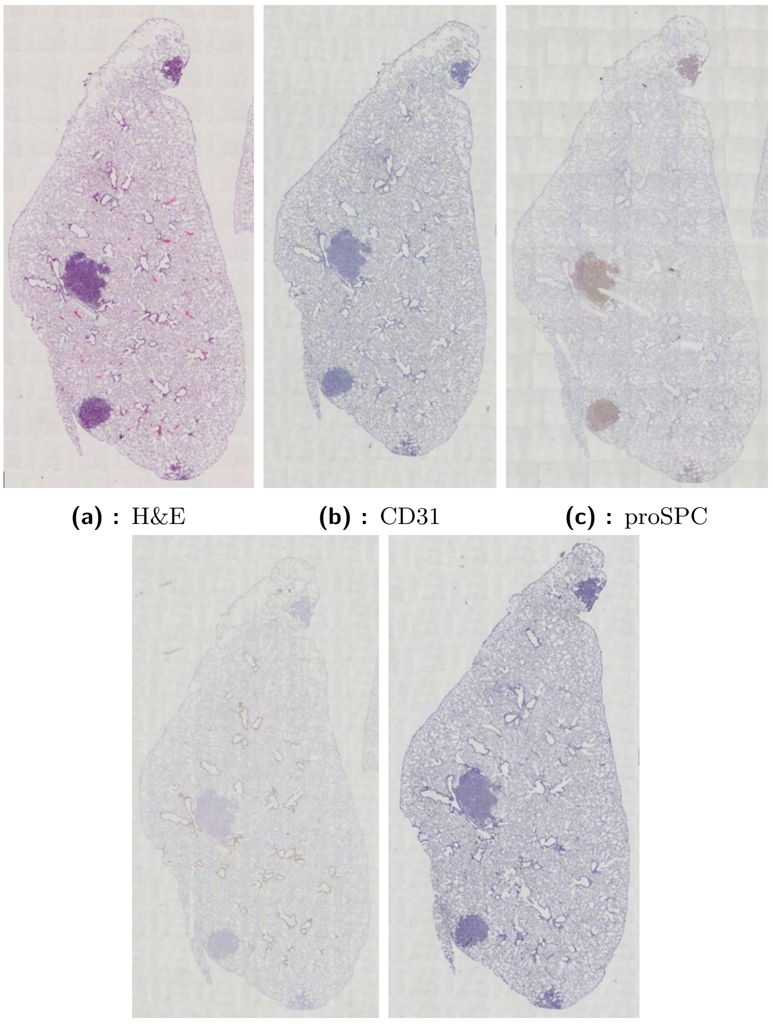
- **Platelet endothelial cell adhesion molecule (PECAM-1 / CD31)**
  Among vascular cell adhesion molecules, platelet-endothelial cell adhesion molecule has a distinctive feature of being expressed on several of the major cell types associated with the vascular compartment [37]. It is, therefore, ideal for the mediation of the cell-to-cell interactions that involve platelets, leukocytes, or endothelial cells [37].

- **Human epidermal growth factor receptor 2 (HER-2)**
  HER-2 (sometimes called c-erbB-2) oncogene is overexpressed in various human cancers (breast, ovarian, lung, gastric or oral). Its presence in the sample is associated with poor overall survival chances [38].

- **Estrogen receptor (ER)**
  ER is a member of the steroid/nuclear receptor superfamily. It is a ligand-activated enhancer of proteins. The function of the ER is to signal transducers and transcription factors to modulate the expression of target genes after ligand binding [39].

- **Progesterone receptor (PR)**
  PR is a regulator in reproductive tissues of women that controls development, proliferation, and differentiation during pregnancy and reproductive cycle [40]. Same as ER, PR is a member of the nuclear receptor family. The primary purpose of PR is to regulate networks of target gene expression after binding. It also plays a role in the progression of endocrine-dependent breast cancer [41].

The following images show the same sample of mice lung lobes after being stained [9].



(a) : H&E **(b) :** CD31 **(c) :** proSPC

(d) : CC10 **(e) :** Ki67

Figure 2.9: Effects of different types of stains

2.4 Description of images in the ANHIR dataset

Images of the samples are taken from

- **Lung lesions:** The base for images in these sets is the 3μm section which is either stained with H&E, CD31, proSPC, CC10, or Ki67. Three mice lung lesions, either adenoma or adenocarcinoma, were acquired at a magnification of 40× and pixel size of 0.174μm/pixel [9].
- **Lung lobes:** Four images of mice lung lobes were acquired in the same fashion as lung lesions, at a lower magnification of 10× and pixel size of 1.274 µm/pixel [9].

- **Mammary glands:** The base for images in this section is two mice mammary glands stained with H&E, the estrogen, and progesterone receptor antibodies or HER2. The images were acquired in the same way as the mice's lung lobes, with a pixel size of 2.294 µm/pixel [9].

- **COAD:** The **Co**lon **Ad**enocarcinoma sets are made of colon cancers samples, taken at 10x magnification. Samples were stained with either H&E histopathology, hematoxylin or DAB [9].

- **Mouse kidney:** Mouse kidneys are similar to human kidneys and, as such, can be substituted. Samples were stained with periodic acid-Schiff (PAS), smooth muscle actin (SMA), or CD31 [9].

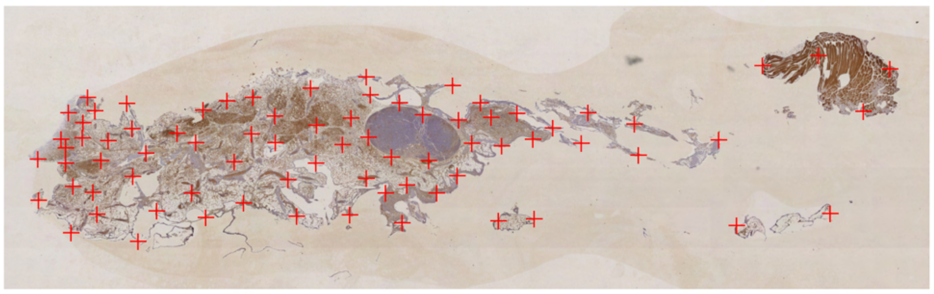
- **Gastric:** Gastric samples in the ANHIR dataset are from patients with a histologically verified gastric adenocarcinoma. Samples were stained with H&E, CD4, CD8 CD68, or CD1a or used for immunophenotyping [9].

- **Human breast:** Samples in these sets were made from the 3μm section of blocks stained with H&E immunohistochemistry(IHC) with antibodies against ER, PR, and HER2 [9].

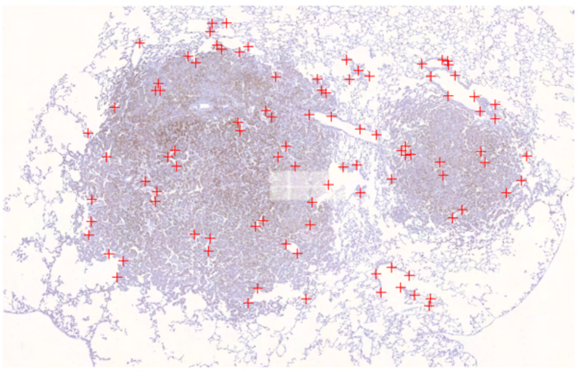
- **Human kidney:** 3μm sections were cut from glomerulopathies blocks and stained with H&E, PAS, Masson, and Methane [9].

2.4.1 Image annotation

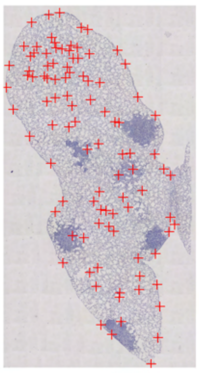
Images are annotated with landmarks signifying important structures in the tissue samples, with 86 landmarks per image on average. The work was done by nine annotators, with each set taking around 2 hours. At least two different annotators annotated each image. Distances between the landmarks placed by annotators were, on average, 0.05% of the image diagonal [9].



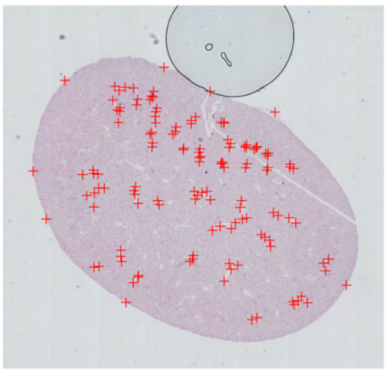
(a) : Mammary glands



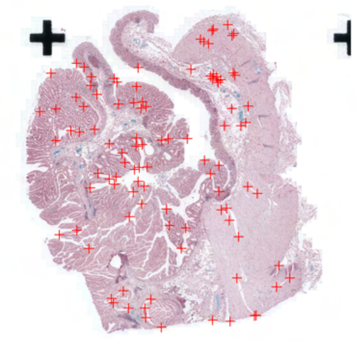
(b) : Lung lesion



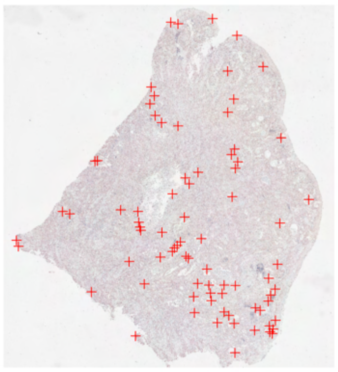
(c) : Lung lobes



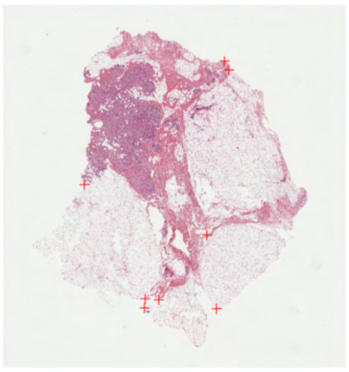
(d) : Mice kidney



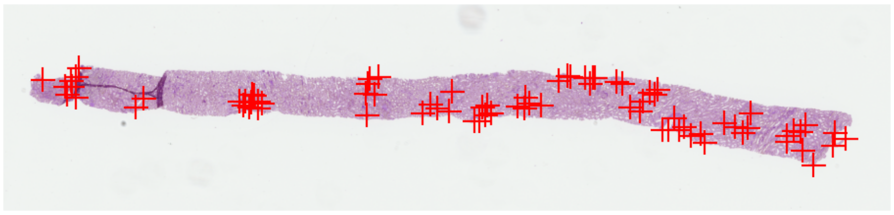
(e) : COAD



(f) : Gastric



(g) : Human breast



(h) : Human kidney

Figure 2.10: Annotated tissues sample from the ANHIR [9] dataset (landmarks are shown as red crosses)

Chapter 3

Data description

For the training of deep neural networks, vast quantities of data are required. We, therefore, use several different datasets, both with and without ground truth optical flow.

Obtaining large datasets of medical images for the training is often difficult due to the privacy issues or lack of annotations done by professionals.

3.1 Flying Chairs

Flying Chairs is a synthetic dataset created in 2015 to train the Flownet architecture [42]. 964 publicly available images of mountains, cities, and landscapes with the resolution of 1024×768 were used for the background, each cropped into a shape of 512×324. The foreground images comprise 809 chairs, with 62 views per chair (with some of the similar images removed to avoid overfitting) [42].

Motion is generated by randomly sampling 2D affine transformation parameters and transforming the chairs relative to the background. The original and the target images are obtained with ground truth optical flow and occlusion regions. Parameters of the affine transformations are adjusted to more closely matched values seen in the MPI-Sintel dataset. 22 872 image pairs and flow fields were generated in total (22 232 train / 640 test) [42].

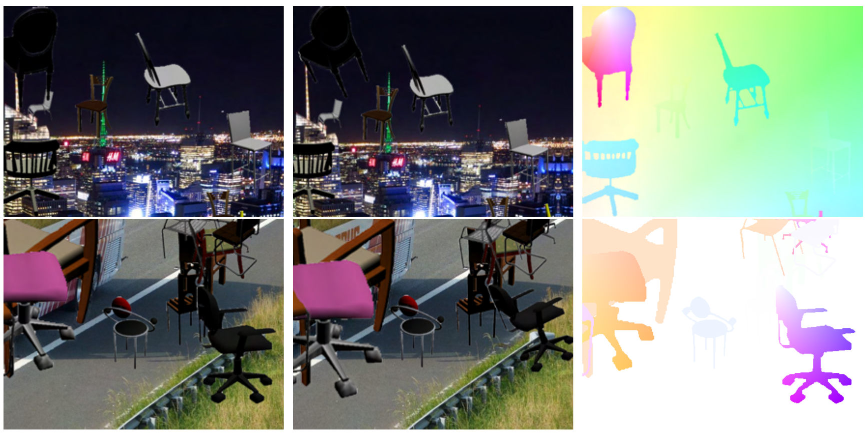


Figure 3.1: Flying Chairs dataset, taken from [42]

3.2 MPI-Sintel

The MPI-Sintel dataset is based on a short open-source animated movie created in Blender with the same name (see `https://youtu.be/eRsGyueVLvQ`) [43]. The open-source nature of this project enabled access to the 3D graphics elements used to create the movie. This allows for the creation of several different types of images, based on the complexity of the rendered scene, ranging from a simple albedo pass, with no shadows rendered and constant albedo over time, to a clean pass, with illumination including shading and specular reflection to a final pass, with all the after-effects, such as blur due to depth of field or atmospheric effects, resembling the original movie.

MPI-Sintel is formed by 35 different image sequences, most around 50 frames long, with corresponding ground truth optical flow. 1628 images were generated, with 1064 belonging to the training set and 564 to the validation set. Images are rendered at a resolution of 1024×436. In addition, unmatched pixels, which were previously invisible due to occlusion, are provided [43].

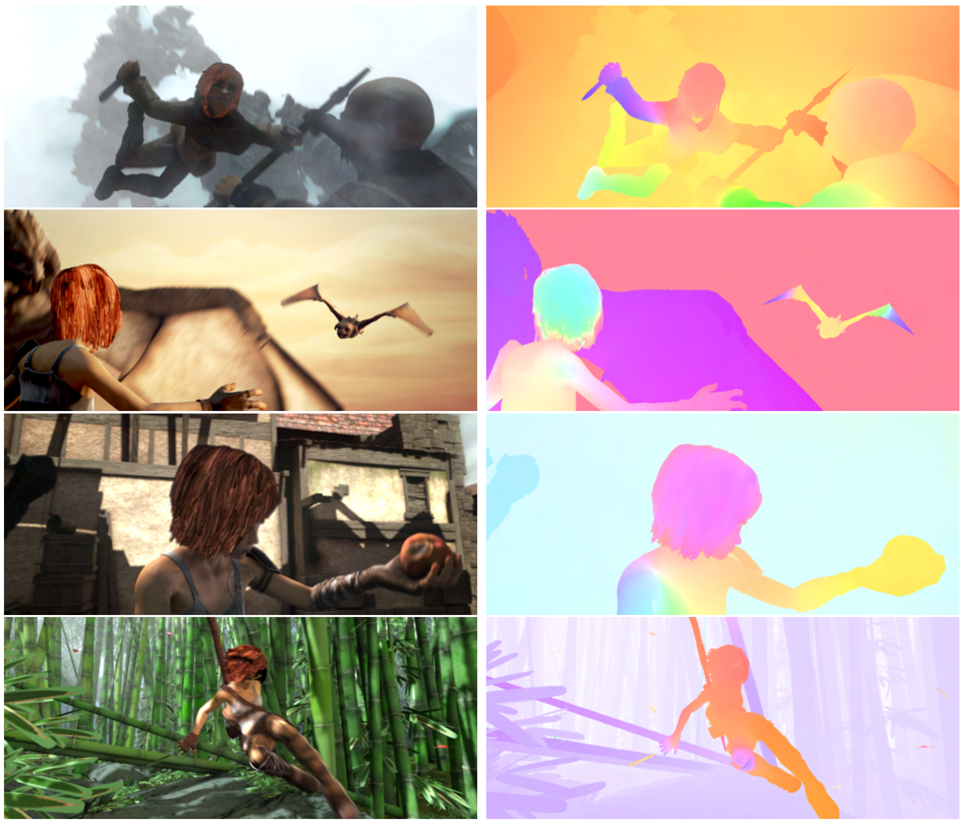


Figure 3.2: MPI-Sintel dataset, taken from [43]

The MPI-Sintel dataset is one of the most commonly used optical flow datasets for the initial training of the networks and a commonly used benchmarking tool.

3.3 ANHIR dataset

The ANHIR dataset is formed by histological images at a microscopic scale that were stained. It contains 49 different sets, with 3 to 9 images per set (average of 5 per set) [9]. In total, there are 355 images with 18 different stains. Each image can be registered with all other images in their respective set, generating 481 image pairs, which are split into 230 training and 251 testing pairs[1] [9]. Within each of the 49 sets, images correspond to the same slice of the same tissue sample [9].

Examples of the images in the ANHIR [9] dataset are shown in full in the section 2.4.

Name	Scanner	Magnitude	μm/ pixel	Avg. size [pixels]	# of sets	# train	# test
Lung lesions	Zeiss	40x	0.174	18k x 15k	3	30	0
Lung lobes	Zeiss	10x	1.274	11k x 6k	4	40	0
Mammary glands	Zeiss	10x	2.294	12k x 4k	2	38	0
Mouse kidney	Hamam.	20x	0.227	37k x 30k	1	15	18
COAD	3DHistech	10x	0.468	60k x 50k	20	84	153
Gastric	Leica	40x	0.2528	60k x 75k	9	13	40
Human breast	Leica	40x	0.253	65k x 60k	5	5	20
Human kidney	Leica	40x	0.253	18k x 55k	5	5	20

Table 3.1: Data summary of the ANHIR dataset, taken from [9]

3.4 Dataset comparison and usage

Flying Chairs and MPI-Sintel were explicitly created to train deep neural networks. Movement in the scenes is mostly rigid, with discontinuities occurring due to occlusion by another object. The color of objects generally does not change. Maximum movement is mostly just a fraction of the scene.

In contrast, images in the ANHIR dataset are different both in shape and color, with maximum displacements of even more than half of the image size. Discontinuities occur due to folding or tearing the tissue, where huge areas suddenly go missing, whereas, in the other two mentioned datasets, the disappearance would have been more sudden.

Each of the three datasets has its unique place in the training of our network. While Flying Chairs or MPI-Sintel have little in common with the histological images, their accurate ground truths with sometimes more superficial scenes are an ideal starting point for the training and, as such, are used for pre-training of our network.

ANHIR dataset does not contain necessary dense ground truth information for learning the optical flow using neural networks since there is no motion. Therefore it needs to be created synthetically. Another issue we are facing is the relatively small size of the dataset, which forces us to use additional data augmentation methods.

[1]The lung lesions and lung lobes datasets were previously used in [10], and were made entirely public. Therefore, they are used for training only.

Name	Pairs total	Pairs train	Pairs test	Image size	Ground truth
Flying Chairs	22872	22232	640	512×324	Yes
MPI-Sintel	1628	1064	564	1024×436	Yes
ANHIR	481	230	251	5k-60k	No[2]

Table 3.2: Comparison of the datasets used for the training

3.5 Evaluation metrics

Before talking about the performance of different methods, we need to define the metrics. In this paper, we will be using metrics used in the ANHIR challenge [9] as well as several other commonly used ones.

The main distinction between the two is what they measure. Measures used in the ANHIR use the landmarks in each of the images, while the other measure image differences in a more general sense.

3.5.1 EPE

End-point-error is a simple metric which coincides with a pixel-wise L2 distance $\|(\cdot)\|_2$ of either the flow, or the landmarks themselves. For optical flow, we are interested in the mean value. It is defined as

$$\mathrm{EPE} = \frac{1}{N} \sum_{\mathbf{x} \in \mathbf{f}} \|\mathbf{f}_{gt}(\mathbf{x}) - \mathbf{f}_o(\mathbf{x})\|_2, \tag{3.1}$$

where $\mathbf{f}_{gt}$ is the ground truth optical flow, $\mathbf{f}_o$ is the predicted flow and $\mathbf{x}$ is a given pixel in the image.

The definition changes slightly for the landmarks and becomes

$$\mathrm{EPE}(k) = \|l_1(i) - l_2(k)\|_2, \tag{3.2}$$

where $l_{(\cdot)}$ is the landmarks pair k.

3.5.2 rTRE

rTRE, or relative target registration error is the end-point-error, divided by the image diagonal as

$$\mathrm{rTRE} = \frac{\mathrm{EPE}}{d}, \tag{3.3}$$

where d is the diagonal computed as $\sqrt{x^2 + y^2}$. Since the images in the ANHIR dataset vary widely in their size, this enables us to normalize the values and compare different sets.

[2]Images are annotated with landmarks, however are missing ground truth for dense registration

3.5.3 MrTRE

Median rTRE will be the main criterion (represented by μ in equations) which we will use to compare individual image registrations. It is defined as

$$\mu^{i,j}(m) = \operatorname*{median}_{l \in L^i} \quad \mathrm{rTRE}_l^{i,j}(m) \tag{3.4}$$

where L is the set of landmarks, which can be found in both images and m is the given method.

3.5.4 Robustness

Robustness is defined as a ratio of the landmarks, which have decreased their distance after the registration (their initial rTRE is higher then their new rTRE). It is defined as

$$R^{i,j}(m) = \frac{|K^{i,j}|}{|L^i|} \tag{3.5}$$

where L^i is the set of all landmarks in the image and $K^{i,j}$ is the subset of the landmarks for which the inequality $\mathrm{rTRE}_l < \mathrm{rIRE}_l$, holds, where rIRE_l is the initial registration error. The mean robustness is then computed over all images as

$$\overline{R}(m) = \operatorname*{mean}_{(i,j) \in \mathcal{T}} \quad R^{i,j}(m) \tag{3.6}$$

3.5.5 Other ANHIR metrics

Several other criteria are computed in the ANHIR challenge, as well as average time required for a registration of a single image pair.

AMrTRE (Average median rTRE)

$$\mathrm{AMrTRE}(m) = \operatorname*{mean}_{(i,j) \in \mathcal{T}} \quad \mu^{i,j}(m) \tag{3.7}$$

MMrTRE (Median of median rTRE)

$$\mathrm{MMrTRE}(m) = \operatorname*{median}_{(i,j) \in \mathcal{T}} \quad \mu^{i,j}(m) \tag{3.8}$$

AMxrTRE (Average maximum rTRE)

$$\mathrm{AMxrTRE}(m) = \operatorname*{mean}_{(i,j) \in \mathcal{T}} \quad \max_{l \in L^i} \quad rTRE_l^{i,j}(m) \tag{3.9}$$

ARMxrTRE (Average rank maximum rTRE)

$$\mathrm{ARMxrTRE}(m) = \operatorname*{mean}_{(i,j) \in \mathcal{T}} \quad \operatorname*{rank}_{m \in \mathcal{M}} \quad \max_{l \in L^i} \quad \mathrm{rTRE}_l^{i,j}(m) \tag{3.10}$$

ARMrTRE

Average rank of median rTRE is the main criterion used to compare results achieved by participants of the ANHIR challenge. It is defined as

$$\mathrm{ARMrTRE}(m) = \operatorname*{mean}_{(i,j)\in\mathcal{T}} \operatorname*{rank}_{m\in\mathcal{M}} \mu^{i,j}(m) \tag{3.11}$$

where $\mathcal{T}$ is the set of image pairs, $\mathcal{M}$ is the set of methods of all the participants.

Chapter 4

Image registration

Image registration is a process of overlaying two or more images of the same object (or related objects) on top of each other. Images could be taken from a different angle, different sensors, different modality, or simply after the object in the image has undergone some form of distortion [44]. One of the common goals in medical imaging is to align tissue samples after their location changes due to inaccurate alignment by the human doctor. As the tissue is usually soft, the transformation will be non-rigid, although a rigid transformation can usually approximate a large part of the transformation. The goal of image registration is to find suitable spatial transformation such that

$$\hat{T} = \underset{T}{\operatorname{argmin}}\, S(I_1, T \circ I_2), \tag{4.1}$$

where I_1 and I_2 are the fixed and the moving image respectively or their features, T are different transformations from some predefined class of transformations and S is a measure of dissimilarity [45]. We use argmax instead of argmin if using SSIM [46] or other similarity based functions.

Basic pipeline for most image registration [44] methods is as follows

1. **Preprocessing** - Images can be filtered to pronounce different features of the image, or their color space transformed to a different one (gray-scale conversion).
2. **Feature detection (or extraction)** - Distinct objects e.g., edges, contours, corners are manually, or automatically detected [45]. Some methods (like ours) instead extract their own features from the image with the help of learnable filters (CNN).
3. **Feature matching** - Corresponding features are matched, and a connection is made. Various descriptor are used for this purpose [45].
4. **Transformation model estimation** - The transformation function is created (in our case, a dense displacement field), which aligns the features of the two images.
5. **Image transformation** - The warping algorithm transforms the image using a transformation function. Sometimes, interpolation is utilized, for the coordinates, which land on non-integer values [45].

4.1 Image similarity functions

To quantify the difference between images, we need a function that allows us to do so. This problem is approachable from many angles, e.g., geometric or probability.

4.1.1 P-norms

One of the simplest ways to quantify the difference between two images is to use a norm of the pixel values. The difference is then

$$L = \frac{1}{N} \sum_{\mathbf{x} \in I_1} \| I_1(\mathbf{x}) - I_2(\mathbf{x}) \|_p, \tag{4.2}$$

where $I_{(\cdot)}$ are the two images in the given point $\mathbf{x}$, and p is the type of norm. Two most used are $p = 1$, which is called sum of absolute differences, and $p = 2$, being sum of squared differences.

4.1.2 Census loss

Census loss utilizes the census transform of the image. This transformation converts the gray-scale image I_g to have nine channels, one for each of the neighbors and the center pixel itself as

$$I_9 = \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}} (I_g \quad | \,\text{in} = 1,\ \text{out} = 9), \tag{4.3}$$

where Conv denotes the convolution operations with given stride s, kernel size 3×3, number of input channels in, and output channels out, the convolution kernel is the identity matrix. The original value is subtracted from the nine-channel image to measure whether the neighborhood has lower or higher intensity values than the center pixel, simply as $I_T = I_9 - I_g$. Value is then normalized as

$$I_N = \frac{I_T}{\sqrt{\epsilon^2 + I_T^2}}, \tag{4.4}$$

where ϵ is some normalization constant. This process is then repeated for the second image, and the normalized Hamming distance is computed for each pixel as

$$d(\mathbf{x}) = \frac{1}{9} \sum_{i=1}^{9} \frac{(I_{N1}^i(\mathbf{x}) - I_{N2}^i(\mathbf{x}))^2}{(I_{N1}^i(\mathbf{x}) - I_{N2}^i(\mathbf{x}))^2 + \epsilon}. \tag{4.5}$$

The overall census loss between the two images is then

$$L_C(I_1, I_2) = \frac{1}{N} \sum_{\mathbf{x} \in I_1} d(\mathbf{x}). \tag{4.6}$$

4.1.3 SSIM

The structural similarity index measure is based on the assumption that the human vision evolved to differentiate between different structures in the scene. Where norm-based dissimilarity function would give us high values, e.g., due to change in brightness, SSIM remains relatively unchanged. The index for the given pair of windows $I_{(\cdot)}$ is given as

$$SSIM(I_1, I_2) = \frac{(2\mu_1\mu_2 + c_i)(2\sigma_{12} + c_j)}{(\mu_1^2 + \mu_2^2 + c_1)(\sigma_1^2 + \sigma_2^2 + c_k)}, \tag{4.7}$$

where $\mu_{(\cdot)}$ is the average of the given window, $\sigma^2_{(\cdot)}$ is the variance, σ_{12} is the covariance and $c_{(\cdot)}$ is used to stabilize division with weak denominator [46].

4.1.4 Cross-correlation

Cross-correlation (CC) matches the image intensities directly (unless CC is used after features have been extracted by a convolutional filter of a CNN or a similar method) [47]. The normalized cross-correlation coefficient in the given point $\mathbf{x}$ can be computed as

$$CC(\mathbf{x}) = \frac{\sum_W (W - E(W))(I(\mathbf{x}) - E(I(\mathbf{x})))}{\sqrt{\sum_W (W - E(W))^2}\sqrt{\sum_{I(\mathbf{x})} (I(\mathbf{x}) - E(I(\mathbf{x})))^2}} \tag{4.8}$$

where W is the window (cropped part of the image, which is used for the cross-correlation), $E(\cdot)$ is the mean and $I(\mathbf{x})$ is the intensity in the given point [47]. This matching method is susceptible to noise and the similarity measure is relatively flat near its maximum [45].

4.1.5 Mutual information

The mutual information (MI), or sometimes called relative entropy, measures statistical dependency between two dataset. MI between two random variables I_1 and I_2, which have been computed from the two images is given as

$$\text{MI}(I_1, I_2) = H(I_2) - H(I_2|I_1) = H(I_1) + H(I_2) - H(I_1, I_2), \tag{4.9}$$

where $H(\cdot) = -E_{(\cdot)}(\log(P(\cdot)))$ is the entropy of the variable and $P(\cdot)$ is the probability of the given variable. Course-to-fine resolution is usually implemented in the so called pyramidal approach (see section 4.2) [48]. MI is maximized, when the two images are aligned, and the information redundancy between the two measured areas is the highest [49].

4.2 Image pyramid

Image pyramids, sometimes called feature pyramids, are sequences of the same image taken at different resolutions. This can be done in multiple ways, either by simply only taking every nth pixel, using a filter, like Gaussian or Laplacian, in order to propagate local information onto a larger area, or interpolating the image [50]. Pyramids generated by CNNs are more flexible, as their learnable kernels allow them to find the ideal filter for the given dataset, as seen in the PWC-Net [51]. Our method uses the interpolated pyramid between the local and global transformation (see section 6.8)and average pooling within the correlation levels (see section 6.6).The primary purpose of the image pyramid in the context of deep neural networks is to find an approximate registration on a lower resolution quickly, which then serves as an initial estimate of the next finer levels.

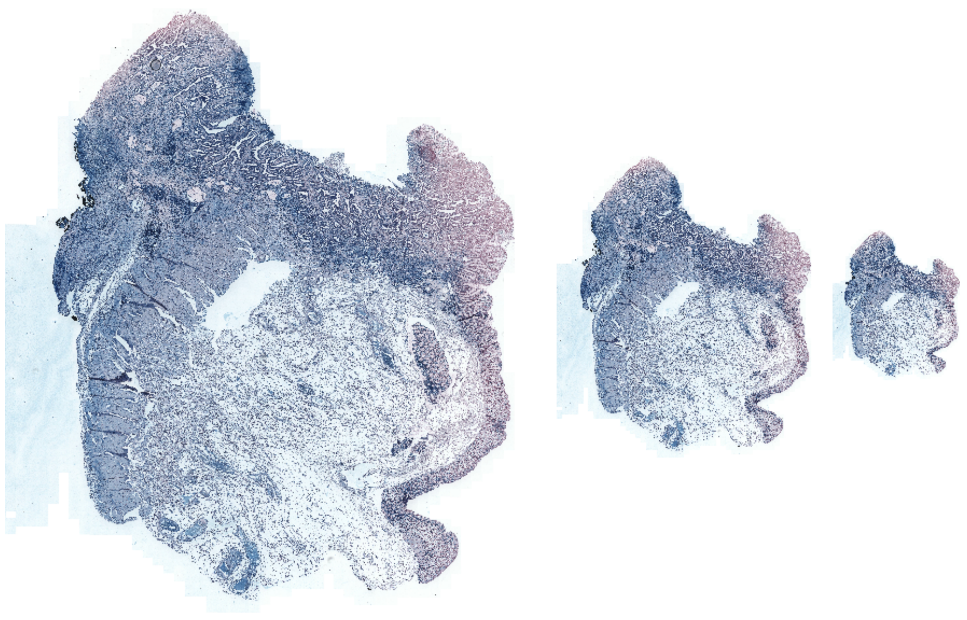


Figure 4.1: Image pyramid created by interpolating to the 1/2 and 1/4 of the original image size

4.3 Transformation models

The transformation models can be divided based on the number of degrees of freedom and whether they are linear or not. The number of control points determines the degree of freedom. Some global models use all control points to estimate one set of parameters used for the entire image. Local models only use several control points for each area and may rely on the global context. However, the higher the number of control points used, the higher the accuracy of the final transformation model. The final transformation can then be approximated with the least-squared fitting, among other methods. This holds true for both the local and global models [45].

In this section, a 2D space is assumed for all transformations. Each point

$\mathbf{x}$ of this space is comprised of 2 coordinates (x, y).

4.3.1 Global registration models

Some of the most frequent global models are bivariate polynomials of low degree.

Similarity transform

Similarity transform (shape-preserving mapping), preserves angles and is determined by two control points. This transformation consist only of rotation, translation and scaling as

$$\begin{aligned} u =& s(x\cos(\theta) - y\sin(\theta)) + t_x \\ v =& s(x\sin(\theta) + y\cos(\theta)) + t_y \end{aligned} \tag{4.10}$$

where s is the scaling factor, θ is the angle of rotation and $t_{(\cdot)}$ is the translation in the given direction [45].

Affine transform

Affine transform provides a more general mapping which is defined by three non-collinear control points [45], defined as

$$\begin{aligned} u =& a_1x + a_2y + t_x \\ v =& b_1x + b_2y + t_y \end{aligned} \tag{4.11}$$

4.3.2 Local registration models

Sometimes the global transformation is not enough to accurately approximate local geometric distortion. In such cases, local methods are utilized.

Radial basis function

Radial basis functions (RBFs) are capable of accurately generating transformations for local geometric distortions. The mapping created by the radially symmetric functions is then

$$u = \sum_{i=1}^{N} c_i g(x, x_i), \tag{4.12}$$

where $g(\cdot)$ is the radial basis function, centered in point x_i, with weight c_i [52].

Thin-plate splines

The most often used RBFs are the thin-plate splines (TPS) [53], where the basis function is defined as

$$g(x, x_i) = \|x - x_i\|^2 \ln(\|x - x_i\|). \tag{4.13}$$

The whole equation is then

$$\begin{aligned} u &= a_0 + a_1 x + a_2 y + \sum_{i=1}^{N} c_i g(x, x_i) \\ v &= b_0 + b_1 x + b_2 y + \sum_{i=1}^{N} c_i g(y, y_i) \end{aligned} \tag{4.14}$$

Chapter 5
Optical Flow

Optical flow approximates the apparent movement of brightness patterns in a series of images (image pair in our case). It can arise from a motion of the object in the scene or relative motion of the object, and the observer [54]. We use the estimated optical flow between image pairs to find a pixel-wise transformation between them. Aside from motion estimation, which is utilized in this thesis, optical flow can also perform motion detection, object segmentation (due to discontinuity of the flow at the borders of the object), motion compensated encoding, or stereo disparity measurement [55].

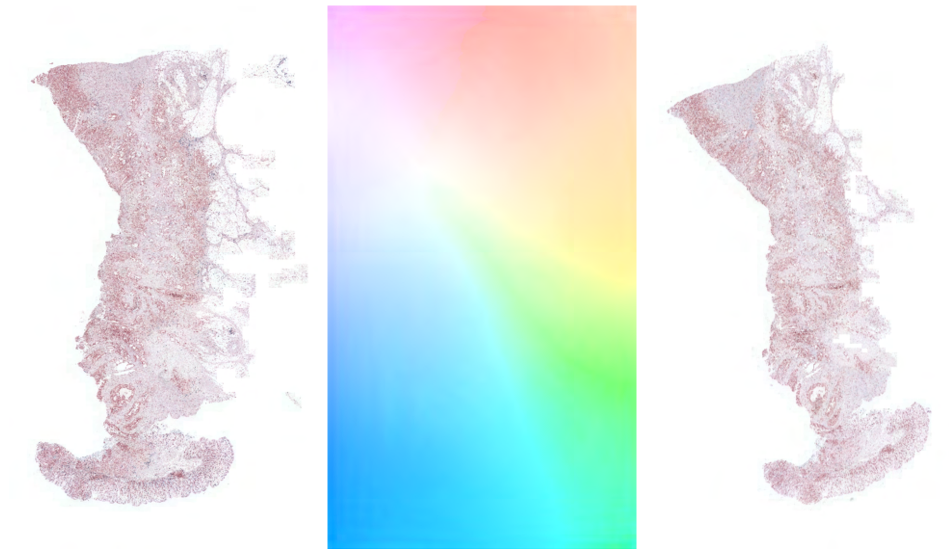


Figure 5.1: Image pair and the corresponding optical flow estimation

Optical flow could be computed locally for each point without additional constraints. However, if the local patch of points all have comparable brightness values, flow cannot be determined locally. As with other registration methods, global, or semi-global context is needed for correct estimation [54].

The underlying assumptions behind optical flow are the invariance of the structure brightness, and local homogeneity of motion [54]. The brightness constraint in each point $\mathbf{x}$ in time t is expressed as

$$\frac{\partial I(\mathbf{x},t)}{\partial t} = 0 \tag{5.1}$$

where I is the brightness of the image.

Since we use an image pair instead of an image sequence, we consider the change in t to correspond to the movement between the two images. Furthermore, if talking about optical flow $\mathbf{f}$, we assume the change from the image in the initial time to the one in the changed time.

5.1 Optical flow estimation

This section discusses several methods used for the optical flow estimations and slightly outlines their derivation.

5.1.1 Basic gradient-based estimation

The equation 5.1 can be rewritten, with the use of optical flow as

$$I(\mathbf{x},t) = I(\mathbf{x}+\mathbf{f}(\mathbf{x}),t+1) \tag{5.2}$$

where I is the intensity of the image in the point $\mathbf{x}$, $\mathbf{f}(\mathbf{x})$ is the flow in the given point and t is a time step [11].

Furthermore, we can approximate the second image by a first-order Taylor series. Firstly, we consider 1D case,

$$f_1(x-d) = f_1(x) - df_1^{'}(x) + \mathcal{O}(d^2 f_1^{''}), \tag{5.3}$$

where d denoted the translation and $f_{(\cdot)}$ are the two signals at two time instants. Let $f_2(x) = f_1(x-d)$. If we ignore higher order terms then the second, the approximation for d is obtained as

$$\hat{d} = \frac{f_1(x) - f_2(x)}{f_1^{'}(x)} \tag{5.4}$$

If generalized to the 2D case, the displaced image is approximated as

$$I(\mathbf{x}+\mathbf{f}(\mathbf{x}),t+1) \approx I(\mathbf{x},t) + \mathbf{f}(\mathbf{x})\cdot\nabla I(\mathbf{x}) + I_t(\mathbf{x},t) \tag{5.5}$$

where I_t denotes the temporal partial derivatives of the image I.

5.1.2 Least-squares estimation

In order to constrain the flow estimation, constraints from surrounding pixels are used, with the assumption of a similar movement in the given patch. The minimization of the constraint error is done by least-squares fitting such that

$$E(\mathbf{f}(\mathbf{x})) = \sum_{\mathbf{x}} g(\mathbf{x})[\mathbf{f}(\mathbf{x})\cdot\nabla I(\mathbf{x},t) + I_t(\mathbf{x},t)]^2, \tag{5.6}$$

where $g(\mathbf{x})$ is a weighted function that determines the support of the estimator. It is usually a Gaussian in order to increase the weight of the pixels in the center of the patch while reducing the importance of the edge pixels [11]. The problem can be rewritten in matrix form as

$$\mathbf{M}\mathbf{f} = \mathbf{b} \tag{5.7}$$

where $\mathbf{M}$ and $\mathbf{b}$ are

$$\mathbf{M} = \begin{bmatrix} \sum g I_x^2 & \sum g I_x I_y \\ \sum g I_x I_y & \sum g I_y^2 \end{bmatrix}, \quad \mathbf{b} = -\begin{pmatrix} \sum g I_x I_t \\ \sum g I_y I_t \end{pmatrix},$$

where I_x and I_y are the spatial partial derivates of the image I. Estimated flow is then computed as

$$\mathbf{f} = \mathbf{M}^{+} b, \tag{5.8}$$

where $\mathbf{M}^{+}$ is the pseudo-inverse of the matrix $\mathbf{M}$.

5.1.3 Iterative optical flow estimation

The solution of the least-squares estimation provides an optimal solution (non in the sense of the original problem). The Taylor polynomial of the first degree is used. We can use Gauss-Newton optimization [56], where the current estimate of the $\mathbf{f}(\mathbf{x})$ is used to undo the motion and look for the residual motion not accounted for by the original estimate. In our method, we use the GRU [57] for the flow iteration. This loop continues until the residual motion is below an acceptable threshold (or a maximum number of iterations is reached) [58]. The warped image sequence is generated as

$$I^j(\mathbf{x}, t) = I(\mathbf{x} + \mathbf{f}^j(\mathbf{x})\delta t, t + \delta t) \tag{5.9}$$

where δt is the time between consecutive frames [11]. In our problem, the consecutive frames represent gradual change from the first image to the second. If the flow difference is not zero, we can estimate the residual flow as

$$\delta\hat{\mathbf{f}}(\mathbf{x}) = \mathbf{M}^{+}\mathbf{b} \tag{5.10}$$

and the next iteration of the refined optical flow as

$$\mathbf{f}^{j+1}(\mathbf{x}) = \mathbf{f}^j(\mathbf{x}) + \delta\hat{\mathbf{f}}(\mathbf{x}) \tag{5.11}$$

and finally the objective function, which we want to minimize is

$$\begin{aligned} E(\delta\mathbf{f}(\mathbf{x})) &= \sum_{\mathbf{x}} g(\mathbf{x})[I^j(\mathbf{x}, t) - I^j(\mathbf{x} + \delta\mathbf{f}(\mathbf{x}), t + 1)]^2 \\ &\approx \sum_{\mathbf{x}} g(\mathbf{x})[\nabla I^j(\mathbf{x} \cdot \delta\mathbf{f}(\mathbf{x})) + I_t^j(\mathbf{x}, \mathbf{t})]^2 = \tilde{E}(\delta\mathbf{f}(\mathbf{x})). \end{aligned} \tag{5.12}$$

We can see, that $\tilde{E}$ approximates E to second-order. This error diminishes, as $\delta\mathbf{f}$ approaches zero.

5.1.4 Global smoothing

The main advantage of global smoothing is propagating information over large areas of the image. The smoothing enables regions of uniform pixel intensity to receive information about flow estimation from regions where the flow could be estimated more easily. One such function is

$$E(\mathbf{f}(\mathbf{x})) = \int (\nabla I \cdot \mathbf{f}(\mathbf{x}) + I_t)^2 + \lambda(\|\nabla f_x\|^2 + \|\nabla f_y\|^2) dxdy, \tag{5.13}$$

which was proposed in [59]. λ is the regularization parameter. The integral can be split into a discrete approximation, which leads to a large system of linear equations [59].

5.1.5 Other methods of optical flow estimation

Many other methods of flow estimations exist. The problem can be formulated from the standpoint of probability, where the likelihood of the flow corresponding to the actual motion is computed. Another approach is the usage of mixture models and utilization of Expectation–maximization (EM) algorithms. EM algorithms assign each pixel within the given region to a small number of possible flows [11].

5.2 Optical flow visualization

The overall motion could be hard to recognize when viewing optical flow, especially in larger images. Therefore a color-coded system is used (Middlebury color code). Each pixel is assigned a color based on the magnitude and direction of the flow.

Hue of the output color is calculated from the angle of the flow as

$$\alpha = \operatorname{atan2}(f_y, f_x) \tag{5.14}$$

where f_x and f_y are components of flow vector $\mathbf{f}(\mathbf{x})$ and atan2 is the two-argument arctangent. After the angle is calculated, the hue is picked from the standard color wheel, 0° corresponding to the red color, 120° to the green color, and 240° to the blue. Saturation is then computed simply as

$$S = \frac{\|\mathbf{f}(\mathbf{x})\|^2}{\max|\mathbf{f}(\mathbf{x})|} \tag{5.15}$$

With fixed V, of the HSV color space, the corresponding RGB color is generated.

Chapter 6
Methods

Over the past several years, overall progress in developing deep neural networks has progressed rapidly. The flow estimation methods generally have similar architecture, with several different modules responsible for different tasks. As described in Image registration, the vast majority of methods follow the same pipeline, that being feature calculation (usually with the help of a feature pyramid), matching, flow estimation from matched features, and some context network for spreading the flow calculated for each pixel into a surrounding area or areas where the motion cannot be determined uniquely due to the aperture problem.

6.1 FlowNet

FlowNet is one of the first successful neural network architectures to estimate optical flow, developed in 2015 [42]. Its architecture is somewhat similar to a standard U-Net.

Feature extraction

We define features as an output of the convolutional layers computed on the whole image. The first part of the network is responsible for feature extraction, composed of several gated convolutional layers (gated in the sense of a non-linearity), which reduce the image's spatial dimensions. These convolutional layers have a stride of 2, and a larger kernel size of 7 or 5 [42].

$$\begin{aligned}
\mathbf{F}_i^1 &= \underset{(7\text{x}7\,|\,\text{s}=2)}{\text{Conv}}(I_i \quad |\, \text{in} = 3, \quad \text{out} = 64)\\
\mathbf{F}_i^2 &= \underset{(5\text{x}5\,|\,\text{s}=2)}{\text{Conv}}(\mathbf{F}_i^1 \quad |\, \text{in} = 64, \quad \text{out} = 128)\\
\mathbf{F}_i^3 &= \underset{(5\text{x}5\,|\,\text{s}=2)}{\text{Conv}}(\mathbf{F}_i^2 \quad |\, \text{in} = 128, \ \text{out} = 256)
\end{aligned}$$

Conv(x, in, out) denotes the convolutional block, described in B.1, which is a combination of a convolutional layer, with the kernel size and stride written underneath (padding is always used to keep spatial dimensions the same after the convolution), batch norm and a non-linearity (LeakyReLU in this case). The variable x represents the input data and arguments in and out of the number of channels. $I_{(\cdot)}$ are the input images ($i \in \{1, 2\}$) and $\mathbf{F}_i^{(\cdot)}$ are their feature maps, after the convolution. The exact process is repeated for the second image, with the same weights, to find the corresponding set of features in the second image [42].

Feature matching

The matching is either done by a dedicated correlation layer, this version is called **FlowNet-C**, or left for the network to decide, called **FlowNet-S** [42].

Correlation for the FlowNet-C is defined as

$$\mathbf{C}(\mathbf{x_1}, \mathbf{x_2}) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{F}_1^3(\mathbf{x_1} + \mathbf{o}), \mathbf{F}_2^3(x_2 + \mathbf{o}) \rangle,$$

where $\mathbf{F}_{(\cdot)}$ is the feature map patch from the given image. Correlation is computationally expensive, requiring $c \cdot K^2$ operations, where $K = 2k + 1$ and c is the number of channels [42]. However, the correlation is not done for each pair of the features and is limited only to those which position in the grid are closer than some predefined constant m (maximum displacement) [42].

In the FlowNet-S, the correlation block is replaced with a simple convolution, with an additional path to the refinement block.

Additionally, a second path that skips the correlation block is used, which allows the network to use the directly obtained information. This path is created as

$$\mathbf{S} = \underset{(1\text{x}1 \,|\, \text{s}=1)}{\text{Conv}}(\mathbf{F}_1^3 \mid \text{in} = 256,\ \text{out} = 32).$$

Feature pyramid and flow estimation

The output of the correlation block $\mathbf{C}$ and the output of the skip path $\mathbf{S}$ are concatenated and in the channels dimension into a single matrix $\mathbf{J^0}$ [42].

The second part of the network is essentially a feature pyramid. As such, each level is constructed as a combination of two convolutional blocks, where the first one halves the spatial resolution of the feature map. This part of the network looks as

$$\begin{aligned}
\mathbf{J}^1 &= \underset{(3\text{x}3 \,|\, \text{s}=1)}{\text{Conv}}(\mathbf{J}^0 \mid \text{in} = 473,\ \text{out} = 256)\\
\mathbf{J}_0^2 &= \underset{(3\text{x}3 \,|\, \text{s}=2)}{\text{Conv}}(\mathbf{J}^1 \mid \text{in} = 256,\ \text{out} = 512)\\
\mathbf{J}^2 &= \underset{(3\text{x}3 \,|\, \text{s}=1)}{\text{Conv}}(\mathbf{J}_0^2 \mid \text{in} = 512,\ \text{out} = 512)\\
\mathbf{J}_0^3 &= \underset{(3\text{x}3 \,|\, \text{s}=2)}{\text{Conv}}(\mathbf{J}^2 \mid \text{in} = 512,\ \text{out} = 512)\\
\mathbf{J}^3 &= \underset{(3\text{x}3 \,|\, \text{s}=1)}{\text{Conv}}(\mathbf{J}_0^3 \mid \text{in} = 512,\ \text{out} = 1024)\\
\mathbf{J}_0^4 &= \underset{(3\text{x}3 \,|\, \text{s}=2)}{\text{Conv}}(\mathbf{J}^3 \mid \text{in} = 1024,\ \text{out} = 1024)\\
\mathbf{J}^4 &= \underset{(3\text{x}3 \,|\, \text{s}=1)}{\text{Conv}}(\mathbf{J}_0^4 \mid \text{in} = 1024,\ \text{out} = 1024)
\end{aligned}$$

Once all the levels of the pyramid have been constructed, the last one with the lowest spatial resolution but the most dimensions is used to estimate the flow $\mathbf{f}^0$. As before, simple convolution is used.

$$\mathbf{f}^0 = \underset{(3x3\,|\,s=1)}{\text{FlowHead}}(\mathbf{J}^4 \mid \text{in} = 1024,\ \text{out} = 2)$$

This block is made of a single convolution with two output channels, one for each spatial direction, without any non-linearity [42].

This flow is upsampled through a flow deconvolutional block, which consists of a transposed convolution without any non-linearity, doubling its spatial resolution each time, matching the resolution of the previous level of the pyramid. We will denote this upsampled flow as $\mathbf{f}_{up}^{(\cdot)}$.

Furthermore, each level of the pyramid is upsampled in the same way (but with a non-linearity this time) into $\mathbf{J}_{up}^{(\cdot)}$ [42].

The flow prediction is then done in a course-to-fine approach as

$$\begin{aligned}
\mathbf{f}^1 &= \underset{(3x3\,|\,s=1)}{\text{FlowHead}}([\mathbf{J}^3, \mathbf{J}_{up}^4, \mathbf{f}^0] \quad \mid \text{in} = 1026,\ \text{out} = 2)\\
\mathbf{f}^2 &= \underset{(3x3\,|\,s=1)}{\text{FlowHead}}([\mathbf{J}^2, \mathbf{J}_{up}^3, \mathbf{f}^1] \quad \mid \text{in} = 770,\quad \text{out} = 2)\\
\mathbf{f}^3 &= \underset{(3x3\,|\,s=1)}{\text{FlowHead}}([\mathbf{J}^1, \mathbf{J}_{up}^2, \mathbf{f}^2] \quad \mid \text{in} = 386,\quad \text{out} = 2)\\
\mathbf{f}^4 &= \underset{(3x3\,|\,s=1)}{\text{FlowHead}}([\mathbf{J}^0, \mathbf{J}_{up}^1, \mathbf{f}^3] \quad \mid \text{in} = 194,\quad \text{out} = 2),
\end{aligned}$$

where $[(\cdot)]$ denotes the concatenation operation.

Supervision

Loss is estimated separately for each of the four pyramid levels. Ground truth optical flow is downsampled to match the spatial resolution of each flow [42]. This approach makes the backpropagation easier, in contrast to only calculating the loss for the last estimation, as the values do not have to propagate through the entire network, but rather each level receives relevant information directly while still backpropagating through lower levels of the pyramid [42].

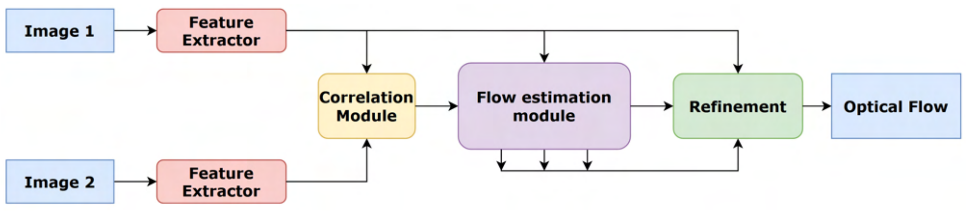


Figure 6.1: FlowNet architecture

6.2 SpyNet

This architecture, developed in 2017, uses a coarse-to-fine pyramid structure to learn the residual flow at each pyramid level [60].

Image pyramid

First, the image is downsampled with the average pooling operation several times (five times was proposed in the original paper [60]). We, therefore, obtain the list of five image pairs at varying resolutions as

$$I_i^k = \underset{(2\text{x}2|\text{s}=2)}{\text{AvgPool}}(I_i^{k-1}).$$

Flow estimation

A basic CNN is created for each of these pairs, with a separate set of weights, which allows each level to better focus on its specific resolution [60]. This network $\mathbf{G}^k$ looks as follows.

$$\begin{aligned}
\mathbf{F}^0 &= [I_1^k, I_w^k, \mathbf{f}^{k-1}] \\
\mathbf{F}^1 &= \underset{(7\text{x}7\,|\,\text{s}=1)}{\text{Conv}}(\mathbf{F}^0 \quad |\ \text{in} = 8, \quad \text{out} = 32) \\
\mathbf{F}^2 &= \underset{(7\text{x}7\,|\,\text{s}=1)}{\text{Conv}}(\mathbf{F}^1 \quad |\ \text{in} = 32,\ \text{out} = 64) \\
\mathbf{F}^3 &= \underset{(7\text{x}7\,|\,\text{s}=1)}{\text{Conv}}(\mathbf{F}^2 \quad |\ \text{in} = 64,\ \text{out} = 32) \\
\mathbf{F}^4 &= \underset{(7\text{x}7\,|\,\text{s}=1)}{\text{Conv}}(\mathbf{F}^3 \quad |\ \text{in} = 32,\ \text{out} = 16) \\
\delta\mathbf{f}^k &= \underset{(7\text{x}7\,|\,\text{s}=1)}{\text{FlowHead}}(\mathbf{F}^4 \quad |\ \text{in} = 16, \quad \text{out} = 2)
\end{aligned}$$

where $\mathbf{F}^{(\cdot)}$ are the feature maps and $\delta\mathbf{f}^k$ is the residual flow estimate. I_1^k is the original downsampled image, while I_w^k is the image I_2^k after it was warped with the flow estimate from the previous level as

$$I_w^k = \text{warp}(I_2^k, \mathbf{f}^{k-1}).$$

The $\mathbf{f}^0$ is set to zero [60].

The residual flow is upscaled with bilinear interpolation and added to the current flow estimate as

$$\mathbf{f}^k = \mathbf{f}^{k-1} + \text{upsample}(\delta\mathbf{f}^k).$$

Each network is only responsible for displacements in the order of several pixels, which simplifies the flow estimation problem. While the first levels need to find bigger relative movements compared to the overall image size, their absolute values remain relatively similar [60].

Supervision

To obtain the ground truth necessary for training each of the networks $\mathbf{G}^k$, the original ground truth optical flow is first downsampled to the desired resolution and subtracted from the GT in the original resolution [60].

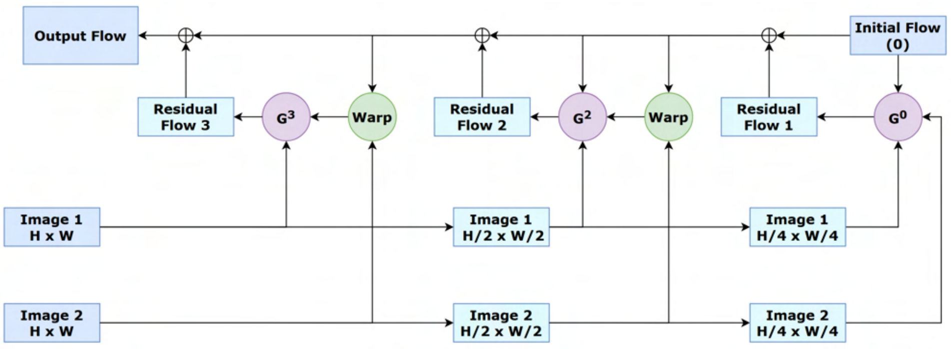


Figure 6.2: SpyNet architecture

6.3 FlowNet 2.0

FlowNet 2.0, developed in 2017, chains several FlowNets (both FlowNet-C and FlowNet-S) into a single network [61]. The first network in the chain is responsible for big displacements and is is fed only the input images I_1 and I_2. All subsequent networks, in addition to the two original images receive the flow $\mathbf{f}^k$, warped image $I_w^k = \text{warp}(I_2, \text{upsample}(\mathbf{f}^{k-1}))$ and the brightest error $\overline{E} = \|I_1 - I_w^k\|_1$ (k denotes the position of the previous network in the chain) [61].

The original paper [61] proposes chaining one FlowNet-C and two FlowNets-S into a single chain (FlowNet2-CSS), which is responsible for more significant displacements. The second chain is created with a modified version of FlowNet-S, with smaller kernels, higher resolutions, and additional convolutions between upsampling layers (FlowNet2-SD). This network is responsible for smaller displacements in the image pair [61].

Finally, the resulting information (flow, flow magnitude, and brightness error) from both the chains is combined inside of a small network, which then outputs the resulting flow at the original resolution [61].

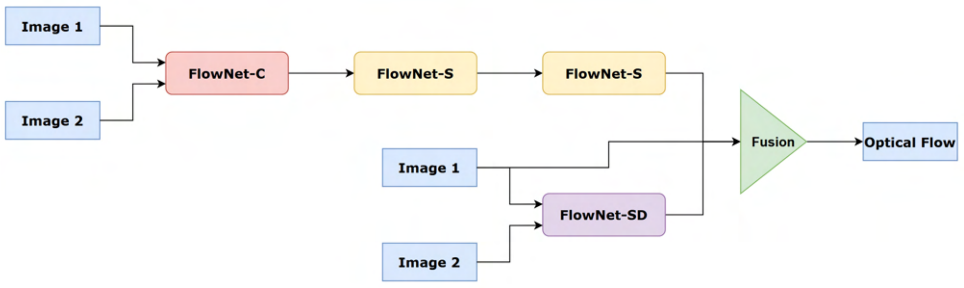


Figure 6.3: FlowNet 2.0 architecture

6.4 PWC-Net

PWC-Net was developed in 2017, the same year as FlowNet 2.0. However, PWC-Net outperforms FlowNet 2.0 architecture while comparatively having a lower number of parameters [51].

Feature pyramid

The first part of the PWC-Net is a feature pyramid extractor. It extracts features similarly to the one seen in the FlowNet architecture [42]. Each level is downsampled to half of the spatial resolution while increasing the number of channels. For the six-level pyramid the number of channels N^i proposed is 16, 32, 64, 96, 128 and 196 [51]. The feature pyramid levels $\mathbf{F}_i^k$ are constructed as

$$\begin{aligned}
\mathbf{F}_i^1 &= \underset{(3\text{x}3\,|\,\text{s}=2)}{\text{Conv}}(I_i \quad |\ \text{in} = N^{i-1}, \quad \text{out} = N^i) \\
\mathbf{F}_i^2 &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}}(\mathbf{F}_i^1 \quad |\ \text{in} = N^{i}, \quad \text{out} = N^i) \\
\mathbf{F}_i^k &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}}(\mathbf{F}_i^2 \quad |\ \text{in} = N^{i}, \quad \text{out} = N^i).
\end{aligned}$$

The feature map is warped with the upsampled flow estimate from the previous level as

$$\mathbf{F}_w^k = \text{warp}(\mathbf{F}_2^k, \text{upsample}(\mathbf{f}^{k-1})),$$

with $\mathbf{f}^0$ being once again set to zero [51].

Feature matching

Correlation $\mathbf{C}^k$ is computed from pairs of features $\mathbf{F}_1^k$ and $\mathbf{F}_2^k$, for which the inequality $\|\mathbf{x}_1 - \mathbf{x}_2\|_\infty < m$ holds, as

$$\mathbf{C}^k(\mathbf{x}_1, \mathbf{x}_2) = \sigma\left(\frac{1}{N}\left((\mathbf{F}_1^k(\mathbf{x}_1))^T \cdot \mathbf{F}_w^k(\mathbf{x}_2)\right)\right),$$

where σ is the LeakyReLU and N the number of correlated elements [51].

Flow estimation

After the correlation, a small CNN is utilized to find optical flow. This network is the same for each of the levels. However, the network weights are not shared for the same reasons as in FlowNet. ResNet-like architecture $\mathbf{R}^k$ consisting of five convolutional blocks is utilized when the original feature map is concatenated with the output of the convolutional layer [51].

The input of this network is

$$\mathbf{x}^0 = [\mathbf{C}^k, \mathbf{F}_1^k, \mathbf{f}_{up}^{k-1}, \mathbf{F}_{up}^{k-1}],$$

and each of the four layers ($i \in \{1, 2, 3, 4\}$) being

$$\mathbf{x}^i = \left[\underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}}(\mathbf{x}^{i-1}), \mathbf{x}^{i-1}\right].$$

The number of channels is not shown, as it varies widely based on the level k and position inside of the network $\mathbf{R}^k$ (see the original paper [51]). The flow, upsampled flow, and upsampled features are created in the same way as the FlowNet architecture.

Context network

For the last level, a context network is utilized. Instead of simply outputting the upsampled flow, the output of a seven-layer CNN called **CN** with increasing dilation is used [51]. This network looks as

$$\begin{aligned}
\mathbf{x}^1 &= \underset{(3\text{x}3\,|\,\text{d}=1)}{\text{Conv}} && (\mathbf{F}^6 |\ \text{in} = IC,\ \text{out} = 128)\\
\mathbf{x}^2 &= \underset{(3\text{x}3\,|\,\text{d}=2)}{\text{Conv}} && (\mathbf{x}^1 |\ \text{in} = 128,\ \text{out} = 128)\\
\mathbf{x}^3 &= \underset{(3\text{x}3\,|\,\text{d}=4)}{\text{Conv}} && (\mathbf{x}^2 |\ \text{in} = 128,\ \text{out} = 128)\\
\mathbf{x}^4 &= \underset{(3\text{x}3\,|\,\text{d}=8)}{\text{Conv}} && (\mathbf{x}^3 |\ \text{in} = 128,\ \text{out} = 96)\\
\mathbf{x}^5 &= \underset{(3\text{x}3\,|\,\text{d}=16)}{\text{Conv}} && (\mathbf{x}^4 |\ \text{in} = 96,\ \ \text{out} = 64)\\
\mathbf{x}^6 &= \underset{(3\text{x}3\,|\,\text{d}=1)}{\text{Conv}} && (\mathbf{x}^5 |\ \text{in} = 64,\ \ \text{out} = 32)\\
\mathbf{f}^{CN} &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{FlowHead}} && (\mathbf{x}^6 |\ \text{in} = 32,\ \ \text{out} = 2),
\end{aligned}$$

where IC is the number of channels of the output of the last network $\mathbf{R}^6$ and d is the dilation (padding is used to preserve the spatial dimension). The purpose of the context network is to propagate flow values to the areas of the images where they could not have been predicted due to a lack of distinguishable features and to remove outliers in the predicted flow [51].

The overall predicted flow of the network is then

$$\mathbf{f} = \mathbf{f}^6 + \mathbf{f}^{CN}.$$

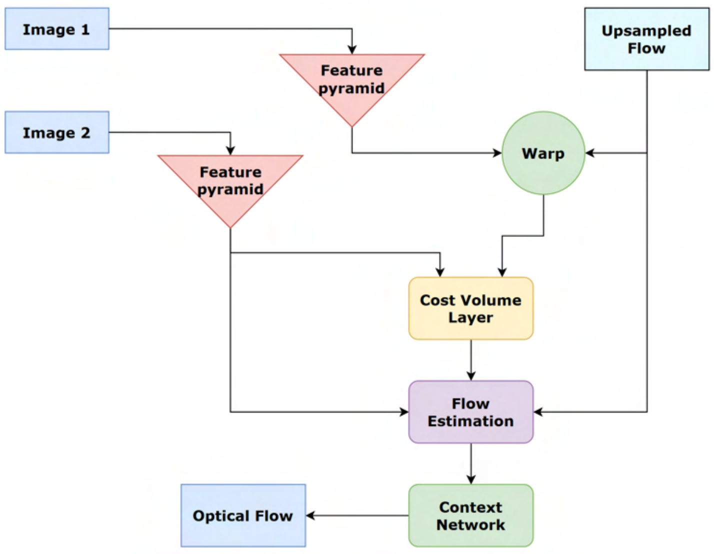


Figure 6.4: PWC-Net architecture

6.5 MaskFlownet

The ambiguity created by occluded areas during warping can create artifacts (doubling of the feature map) [62]. MaskFlownet introduces an asymmetric occlusion-aware feature matching module (AsymOFMM).

AsymOFMM incorporates learnable occlusion mask θ and additional feature tensor μ, which filters useless information after feature warping. Additionally, an extra convolutional layer $\mathcal{D}$ is introduced, which deforms only one of the images [62].

The overall network architecture is similar to a standard PWC-Net. However, instead of refining only the flow at each level of the feature pyramid, occlusion mask θ and feature tensor μ are also computed.

Occlusion-aware mask

For the creation of the mask, an additional convolutional layer is introduced at each level, which outputs a single channel, which is mapped to a $[0, 1]$ range by a sigmoid function and upscaled to match the spatial resolution of the next level of the pyramid, where it is multiplied with the flow prediction [62].

Feature tensor

The feature tensor μ is computed similarly, without being limited to a single channel or the $[0, 1]$ range, and is then added to the predicted flow [62].

Deformable convolution

The deformation is done with a deformable convolutional module [63], which instead of using a standard rectangular kernel that samples the feature map in the specific integer grid, creates its own kernel, which augments the kernel with an offset to its location, allowing the layer to sample even in fractional locations, changing the convolution from B.1 to

$$I_O(i,j) = \sigma\left(b + \sum_{(m,n)\in K} I_I(i+m+\Delta m, j+n+\Delta n)K(m,n)\right), \tag{6.1}$$

where Δm and Δn are the learned offsets of the given location in the kernel [63].

The overall process done by the AsymOFMM at level k can then be expressed as

$$\begin{aligned} I_w^k &= \mathcal{D}^k(\mathrm{warp}(I_2^k, \mathbf{f}_{occ}^k)), \\ \text{where} \quad \mathbf{f}_{occ}^k &= \mathbf{f}^{k-1} \odot \theta^{k-1} \oplus \mu^{k-1} \end{aligned} \tag{6.2}$$

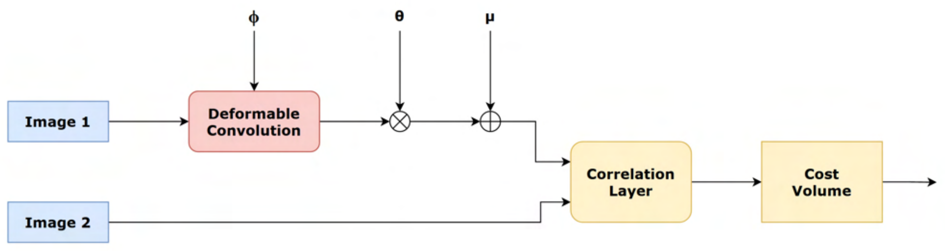


Figure 6.5: AsymOFMM architecture

6.6 RAFT

Recurrent All-Pairs Field Transforms architecture estimates flow at a single high-resolution (while still using the pyramid approach) [57].

Feature extraction

Similar to other architectures, features are extracted at a smaller resolution (1/8) with the feature network (FN). This network consists of a standard input and output convolution and two residual blocks in the middle [57]. FN is then

$$\begin{aligned}
\mathbf{F}_i^1 &= \underset{(7\text{x}7\,|\,\text{s=2})}{\text{Conv}} && (I_i \;\;|\text{ in} = 3,\text{ out} = 64)\\
\mathbf{F}_i^2 &= \underset{(3\text{x}3\,|\,\text{s=2})}{\text{ResBlock}} && (\mathbf{F}_i^1 \mid \text{in} = 64, \text{out} = 96)\\
\mathbf{F}_i^3 &= \underset{(3\text{x}3\,|\,\text{s=2})}{\text{ResBlock}} && (\mathbf{F}_i^2 \mid \text{in} = 96, \text{out} = 128)\\
\mathbf{F}_i &= \underset{(1\text{x}1\,|\,\text{s=1})}{\text{Conv}} && (\mathbf{F}_i^3 \mid \text{in} = 128, \text{out} = 128)
\end{aligned}$$

Context network

The context network (CN) has the identical architecture to FN, with its output being split into two matrixes with different channel count, the first, which remains the same during subsequent updates, mapped into the $[0, \infty]$ range by the ReLU and second, called hidden state, to the $[-1, 1]$ by the tanh [57].

Feature matching and correlation pyramid

Full correlation volume is computed from a dot product of the feature maps as

$$\mathbf{C}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{N}(\mathbf{F}_1(\mathbf{x}_1))^T \cdot \mathbf{F}_2(\mathbf{x}_2). \tag{6.3}$$

For feature maps with dimensions $\mathbb{R}^{H\times W}$, the resulting correlation volume will have dimensions $\mathbb{R}^{H\times W\times H\times W}$. A four-level correlation pyramid is constructed, with each level halving the spatial resolution of the two last dimensions of $\mathbf{C}$ [57].

A correlation lookup operator $L_{\mathbf{C}}$ is defined [57]. This operator generates feature maps by indexing from the correlation pyramid within the given radius r. Given current estimate of the flow f^k, each pixel from the I_1 is mapped to I_2 as $\mathbf{x}_1 = \mathbf{x}_2 + \mathbf{f}^k(\mathbf{x_1})$. A grid $\mathcal{N}$ is created at each point, such that

$$\mathcal{N}(\mathbf{x}_2) = \{\mathbf{x}_2 + \mathbf{dx} \mid \mathbf{dx} \in \mathbb{Z}^2, \|\mathbf{dx}\|_1 < r\} \tag{6.4}$$

A lookup is performed at all the levels of the pyramid, with each level generating a single feature map [57].

Flow estimation

First, the coordinate systems Cr_1^0 and Cr_2^0 are initialized by creating two meshgrids [57]. Flow is computed as difference of these coordinates as

$$\mathbf{f}^k = Cr_1^k - Cr_2^k. \tag{6.5}$$

The operator takes flow, correlation volume, and hidden state and outputs updated hidden state and residual flow $\delta\mathbf{f}^k$. The residual flow is used to transform the coordinate system as

$$Cr_1^k = Cr_1^{k-1} + \delta\mathbf{f}^{k-1}, \tag{6.6}$$

which in turn updates the current estimate of the optical flow as

$$\mathbf{f}^k = \mathbf{f}^{k-1} + \delta\mathbf{f}^{k-1}. \tag{6.7}$$

The operator is designed to mimic the steps of an optimization algorithm and converge to the fixed point $\mathbf{f}^k \rightarrow \mathbf{f}^*$, where $\mathbf{f}^*$ is the optimal solution [57].

Update operator

The update operator can be split into four distinct parts. Firstly, a *motion encoder*, which takes the correlation volume $\mathbf{C}^k$ and the flow estimate $\mathbf{f}^k$ and outputs motion features $\mathbf{MF}$ as

$$\begin{aligned}
\mathbf{C}^i &= \underset{(1\text{x}1\,|\,\text{s}=1)}{\text{Conv}} \left(\mathbf{C}^k \mid \text{in} = CP, \text{out} = 256\right) \\
\mathbf{C}^j &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}} \left(\mathbf{C}^i \mid \text{in} = 256, \text{out} = 196\right) \\
\\
\mathbf{f}^i &= \underset{(7\text{x}7\,|\,\text{s}=1)}{\text{Conv}} \left(\mathbf{f}^k \mid \text{in} = 2, \text{out} = 128\right) \\
\mathbf{f}^j &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}} \left(\mathbf{f}^i \mid \text{in} = 128, \text{out} = 64\right) \\
\\
\mathbf{MF} &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}} \Big([\mathbf{f}^j, \mathbf{C}^j] \mid \text{in} = 192 + 64, \text{out} = 128 - 2\Big) \\
\mathbf{MF} &= [\mathbf{MF}, \mathbf{f}^k]
\end{aligned}$$

The core component of this operator is a gated activation unit (GRU) [57], which structure looks as

$$\begin{aligned}
\mathbf{x}_t &= [\mathbf{MF}, CF] \\
z_t &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}}([h_{t-1}, \mathbf{x}_t] \mid \text{in} = 128 + 128 + 196, \text{out} = 128) \\
r_t &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}}([h_{t-1}, \mathbf{x}_t] \mid \text{in} = 128 + 128 + 196, \text{out} = 128) \\
\tilde{h_t} &= \underset{(3\text{x}3\,|\,\text{s}=1)}{\text{Conv}}([\sigma(r_t) \odot h_{t-1}, \mathbf{x}_t] \mid \text{in} = 128 + 128 + 196, \text{out} = 128) \\
h_t &= (1 - \sigma(z_t)) \odot h_{t-1} + \sigma(z_t) \odot \tanh(\tilde{h_t})
\end{aligned}$$

where CF are the context features and h_{t-1} is the previous hidden state[57].

Residual flow is computed from the updated hidden state, with the flow head module [57]. Lastly, a flow mask is created as an output from two convolutional layers [57].

Supervision

During each update, the upsampled flow estimate is saved for the training. Similar to FlowNet, creating the loss function, not only for the network's final output, allows the gradient to propagate easier. However, the weight of the earlier flows is reduced (see section B.5).

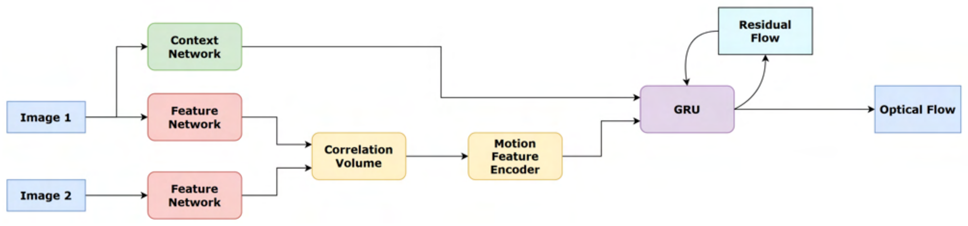


Figure 6.6: RAFT architecture

6.7 GMA

Global Motion Aggregation (GMA) architecture is based on the RAFT while introducing a new module, which creates additional information for the GRU cell [64].

This module, based on the transformer architecture, is used to find long-range dependencies between image pixels and motion features, effectively bundling them together and allowing the network to choose between global and local motion features [64]. The GMA module computes the feature vector update as an attention-weighted sum of the projected motion features [64] and is given as

$$\mathbf{\hat{y}_i} = \mathbf{y_i} + \alpha \sum_{j=1}^{N} FS(\theta(\mathbf{x_i}), \phi(\mathbf{x_j}))\sigma(\mathbf{y_j})$$

where α is a learned scalar parameter, θ, ϕ, and σ are the projection functions of the query, key, and value vectors (each with their respective learnable projection matrix). FS is the similarity attention function defined as

$$FS(\mathbf{a}_i, \mathbf{b}_j) = \frac{\exp\left(\mathbf{a}_i^T \mathbf{b}_j \sqrt{D}\right)}{\sum_{j=1}^{N} \left(\mathbf{a}_i^T \mathbf{b}_j \sqrt{D}\right)}. \tag{6.8}$$

The projection functions for the query, key and value vectors are

$$\begin{aligned} \theta(\mathbf{x}_i) &= \mathbf{W}_{\text{qry}} \mathbf{x}_i, \\ \phi(\mathbf{x}_i) &= \mathbf{W}_{\text{key}} \mathbf{x}_i, \\ \sigma(\mathbf{y}_i) &= \mathbf{W}_{\text{val}} \mathbf{y}_i, \end{aligned} \tag{6.9}$$

where $\mathbf{W}_{\text{qry}}$, $\mathbf{W}_{\text{key}}$ and $\mathbf{W}_{\text{val}}$ are the learnable parameters of the GMA module [64].

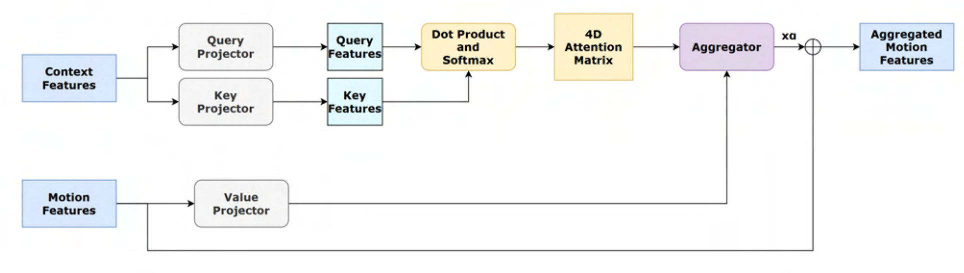


Figure 6.7: GMA module architecture

6.8 Our method

Our method utilizes a course-to-fine approach with two distinct modules. The first one is responsible for global and the second for local registration. The global and local modules use the GMA network as their respective backbone. However, weights are not shared between the two.

The global network estimates flow on a downsampled image, which is then interpolated to match the original resolution and is used to warp the original image alongside the landmarks.

The failure state of the global registration is determined by census loss before and after registration.

The warped image and landmarks and the target image and landmarks are then used as the input of the local module.

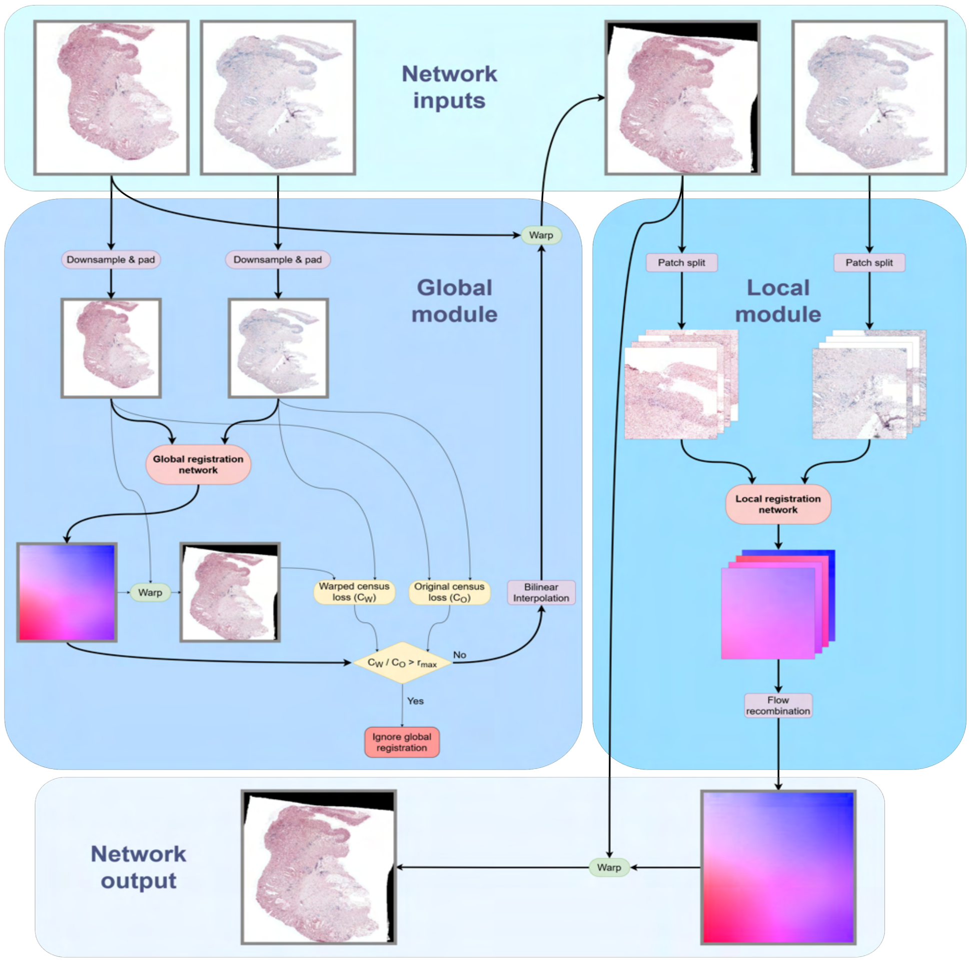


Figure 6.8: Overall registration process of our method

6.8.1 Global registration

Images are downsampled to a smaller size and padded so that their size is the same and are in shape, which the network can process (side length divisible by 8).

The global network is trained with a higher weight of the smoothness loss (see section 8.5) function and lower weight of the ground truth optical flow to force the network to register significant movements and ignore smaller displacements. Images used for training are smaller in size but with bigger relative movement to force the global network to learn its desired function.

Optical flow $\mathbf{f}_g$ from the global network is first bi-linearly interpolated to the original size, and the values are scaled based on the change in the said size as

$$\mathbf{f} = [r_x, r_y] \cdot \underset{(x_o, y_o)}{\text{interpolate}}(\mathbf{f}_g) \tag{6.10}$$

where $r_x = x_o/x_g$ and $r_y = y_o/y_g$ are the ratios of the original image size and the image size used in the global network. The original image, alongside its corresponding landmarks, is warped with the globally found optical flow and fed into the local module for further registration.

We have chosen to use the GMA architecture [64] as our global network, as the GMA module makes estimating the global movement easier than the standard RAFT [57]. The primary purpose of the GMA module was to solve issues raised by occlusion. Since extensive parts of the tissue sometimes go missing, either due to tearing or folding, and entire structures disappear due to different stain dyeing, we, therefore, have a use for such a module.

It is also important to note that both the RAFT and GMA networks, which we are using, are not capable of registering images with displacements higher than 256 pixels. This value is based on the receptive field of the correlation module, which creates a 4 level correlation pyramid and the lookup operator with a radius of 4 ($4^4 = 256$, see section RAFT).

6.8.2 Local registration

Due to memory constraints, we cannot register the whole image simultaneously in high resolution. Instead, the image is split into smaller patches, which are then fed into the local network.

We assume that the patch is big enough to have both the original and target features inside it. However, a higher weight is given to the flow predictions from the middle of the patch.

The resulting flow is then recombined into the original resolution as

$$\mathbf{f}(\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i(\mathbf{x})\mathbf{f}_i(\mathbf{x})}{\sum_{i=1}^{N} w_i(\mathbf{x})}, \tag{6.11}$$

where $w_i(\mathbf{x})$ is the weight mask of the given flow patch $\mathbf{f}_i(\mathbf{x})$ in point $\mathbf{x}$.

6.9 Other methods in the ANHIR challenge

The ANHIR paper [9] describes 13 different methods, seven created by the participants and six baseline methods provided by the organizers.

All of the methods in the ANHIR paper submitted by the participants use grayscale images and an affine/rigid pre-alignment. Our method, however, uses color images, and the pre-alignment is done non-rigidly as well. The methods provided by the organizers use B-splines as the non-linear transformation, while the participants use a wider variety of different, primarily dense transformations.

Method	Grayscale	Exhaustive search	Affine/rigid pre-alignment	Feature points	Non-linear transformation	Criterion	Optimization
UA	•		•		dense moving mesh	MI	gradient
TUNI	•		•	•	virtual springs	NCC	robust linear
CKVST	•		•		B-splines	NCC	L-BFGS
MEVIS	•	•	•		dense	NGF	L-BFGS
AGH	•	•	•	•	dense	various	various
UPENN	•	•	•		various	NCC	L-BFGS, gradient
TUB	•		•		dense	NCC	CNN
bUnwarpJ*	•				dense	SSD	LM
RVSS*	•		•	•	B-splines	SSD	LM
NiftyReg*	•		•		B-splines	NCC, MI	conjugated-gradients
Elastix*	•		•		B-splines	NCC	ASGD
ANTs*	•		•		B-splines	NCC, MI	L-BFGS
DROP*	•		•		B-splines	SSD	discrete optimization
Our method					dense	various	CNN

Table 6.1: Methods in the ANHIR challenge, (* - organizer methods) taken from [9]

Further reading on the specifics of each of the methods is available in the ANHIR paper [9].

Chapter 7

Simulated ANHIR dataset

The goal of our thesis is to register images in the ANHIR dataset [9], however, this dataset does not contain the necessary ground truth for optical flow estimation, and as such, we need to create our own, alongside with images generated by said flow. Contained in the dataset are images in varying resolutions, ranging from images that have several hundred to tens of thousands of pixels in width and height.

The high-resolution images are downsampled to a more manageable size of around a few thousand pixels.

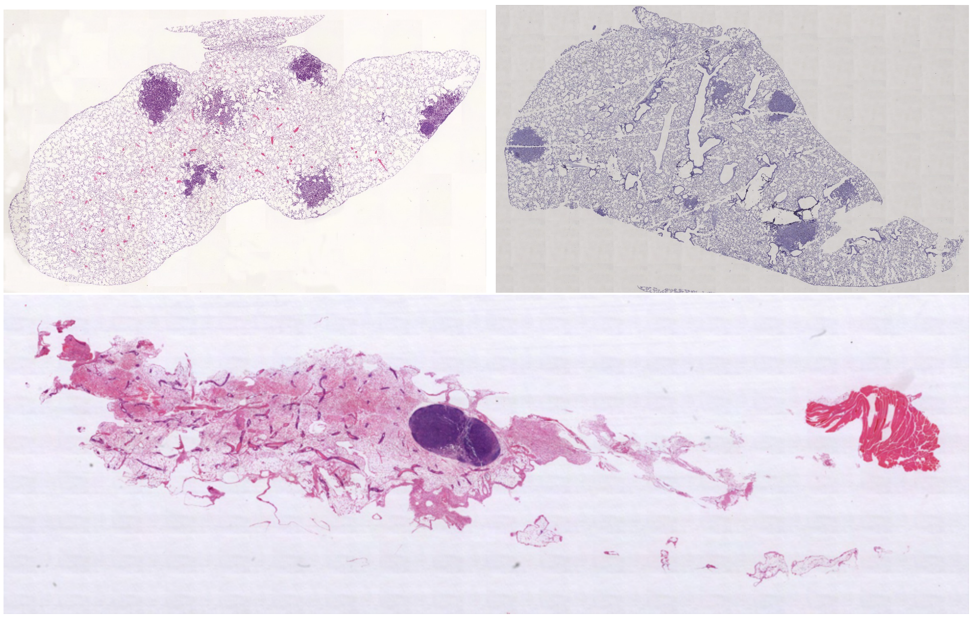


Figure 7.1: Examples of the images in the ANHIR dataset

The training triplets (original image, target image, and corresponding optical flow) are generated in several different ways. Either the target image is created from the original image as a combination of several different transformations, which are translated to flow or when the image has a high number of landmarks, the flow is directly generated from the coordinate difference of said landmarks.

7.1 Affine approximation of the transformation

Transformations done to the images in the ANHIR dataset can be split into rigid and non-rigid parts. We have chosen to approximate the rigid part as an affine transformation.

To find the transformation matrix, least-squares fitting is utilized. We try to find a 3x3 matrix **A** (represented by a 6 item matrix in the optimization task), which corresponds to the affine transformation and solves the equation

$$\mathbf{XA} = \mathbf{b} \tag{7.1}$$

in the least-squares sense as

$$\mathbf{A} = \mathbf{X}^{+}\mathbf{b}. \tag{7.2}$$

Matrix **X** is generated from the landmarks of the first image as

$$\mathbf{X} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \tag{7.3}$$

where $x_{(\cdot)}$ and $y_{(\cdot)}$ are the coordinates of the given landmark. The matrix **b** is constructed from the landmarks of the second image as

$$\mathbf{b} = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix}. \tag{7.4}$$

We calculate the matrix **X** for all the image pairs, then the mean and the variance of each of the six parameters needed for the affine transformation are calculated. This transformation, however, does not describe pair-wise and higher-order dependencies, and values vary drastically between different sets of images.

Values calculated from the ANHIR dataset are

$$\begin{bmatrix} 0.993 \pm 0.0138 & 0.0221 \pm 0.0448 & 0.00673 \pm 0.00553 \\ -0.0210 \pm 0.0427 & 0.989 \pm 0.0136 & 0.0122 \pm 0.00595 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.5}$$

Values corresponding to the translation have been scaled to correspond to the fraction of the image, instead of the absolute pixel distance, to facilitate the different image shapes better.

The original landmarks $l_O(i)$ in the ANHIR dataset have a mean rTRE (see section 3.5) of roughly 0.0655, while landmarks after the affine transformation $l_A(i)$, calculated for each of them as

$$l_A(i) = \mathbf{A}\, l_O(i) \tag{7.6}$$

have a mean rTRE of 0.00495, meaning roughly 92.5% of the transformation is rigid.

We sample from a random normal distribution with the given means and variances corresponding to each of the six parameters from the matrix 7.5. The corresponding synthetic affine flow is then computed from a difference of coordinates before and after affine transformation as

$$\mathbf{f}(\mathbf{x}) = C_A(\mathbf{x}) - C_O(\mathbf{x}) \tag{7.7}$$

where C_O is the original coordinate grid and C_A is the transformed coordinate grid of the image.

7.2 Flow estimation from landmarks

Each image has a corresponding list of landmarks, with their coordinates as (x, y). We denote landmarks for the original image as l_O and l_T for the target image. We, therefore, have an accurate estimate of the flow $\mathbf{f}$ in several points in the image, computed as

$$\mathbf{f}(l_O(i)) = l_O(i) - l_T(i). \tag{7.8}$$

However, we need to extrapolate this sparse flow onto the whole image. For the extrapolation, we use k of the nearest landmarks (usually 5). Their distance being

$$d(\mathbf{x}, i) = ||\mathbf{x} - l_O(i)||_2 \tag{7.9}$$

with $\mathbf{x}$ being a pixel coordinate and i the index of the corresponding landmark. The optical flow estimate is then computed as a weighted average of the k points, computed as

$$\mathbf{f}(\mathbf{x}) = \frac{\sum_i d(\mathbf{x}, i)^{-2}\mathbf{f}(l_O(i))}{\sum_i d(\mathbf{x}, i)^{-2}} \tag{7.10}$$

where $i \in k - NN$. We then blur the flow to eliminate any discontinuities on the boundaries between landmarks. Optical flow created with this method has rTRE (see section 3.5) in the range of $0.0005 - 0.0036$ on the landmarks while being relatively smooth in the rest of the image. Other methods, such as creating the flow from the weights of a Gaussian mixture, where each landmark is in the center of 2D Gaussian with some arbitrary variance (see section 7.3), while being smoother, increase the rTRE well above 0.01, which is not acceptable for the direct creation of the flow between images as errors are more than the goal of our registration.

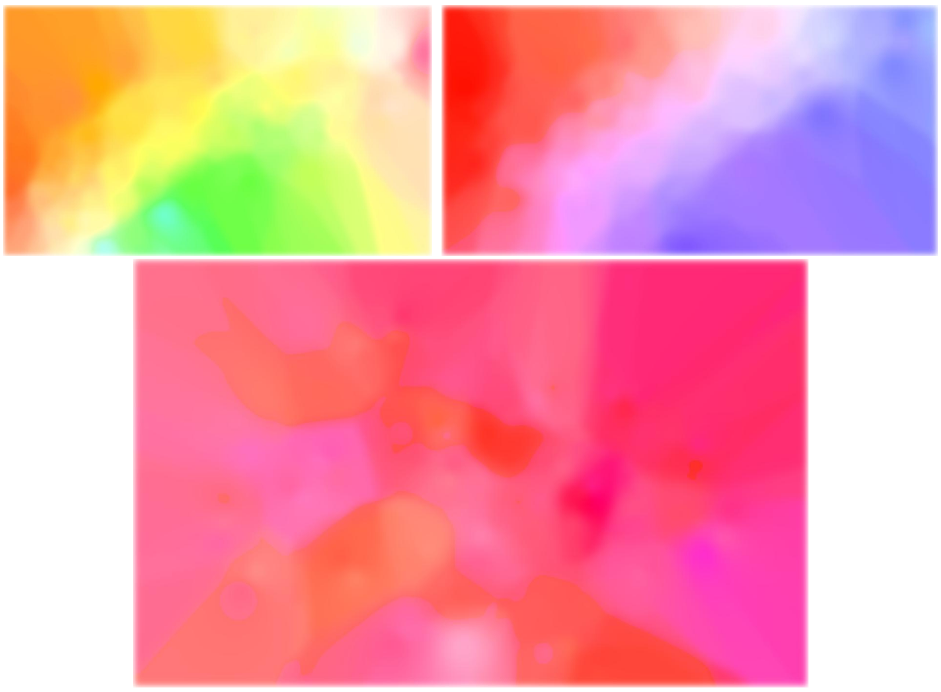


Figure 7.2: Flow generated from the landmarks

7.3 Flow generation from random distribution

For the random generation, a multivariate Gaussian mixture is used. This method is capable of roughly approximating the transformation of images in the ANHIR dataset and, as such, is ideal for the creation of the target image from the original image as we get the exact flow responsible for the transformation. Each pixel $\mathbf{x}$ of the image has some assigned value

$$\mathbf{f}_c(\mathbf{x}) = \sum_{i=1}^{N} D_i \frac{1}{\sqrt{2\pi|\mathbf{\Sigma}_i|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu_i)} \tag{7.11}$$

where $\mathbf{\Sigma}_i$ are the covariance matrixes, μ_i are the centers, $D_i \in \{-1, 1\}$ is chosen randomly and determines the direction of the flow, and N is the number of multivariate Gaussians contributing to the mixture. Since we do not want to create displacement with magnitudes, which are too high, the values are normalized into the $[0, 1]$ range as

$$\mathbf{f}_n(\mathbf{x}) = \frac{\mathbf{f}_c(\mathbf{x}) - \min_x \mathbf{f}_c(\mathbf{x})}{\max_x \mathbf{f}_c(\mathbf{x}) - \min_x \mathbf{f}_c(\mathbf{x})} \tag{7.12}$$

and finally into the $[-m, m]$ range as

$$\mathbf{f}(\mathbf{x}) = m \cdot (2\mathbf{f}_n(\mathbf{x}) - 1) \tag{7.13}$$

where m is the maximum displacement, this process is repeated to obtain the flow for the other direction. The resulting synthetic flows then look as shown in the figure 7.3.

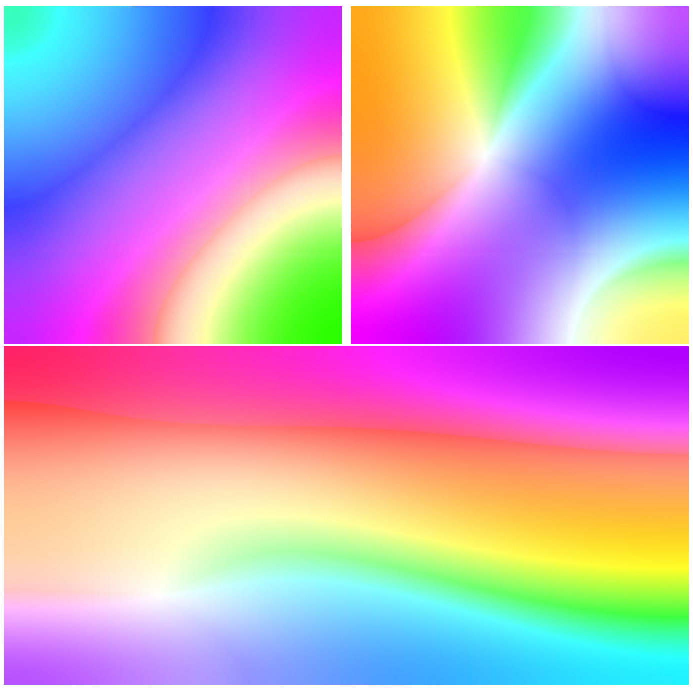


Figure 7.3: Synthetic flow generated from a Gaussian mixture

7.4 Image segmentation

As we utilize several unsupervised loss functions for the fine-tuning, we do not want to penalize the network for incorrect predictions in areas of the image which are of no interest to us. Instead, we only focus on the image's foreground (that being the tissue itself). The segmentation is done naïvely by first filtering the image with a high-pass filter. For this, we use the Ricker (Mexican hat) wavelet, which is defined as

$$\psi(x,y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{7.14}$$

normalization into the $[0,1]$ range as in equation 7.12, and simple thresholding against the mean value of the filtered image as

$$I_O(\mathbf{x}) = \left[I_I(x) > \overline{I_I}\right] \tag{7.15}$$

where $[(\cdot)]$ is the Iverson bracket and $\overline{I_I}$ is the mean value of the image, mapping the output to $\{0,1\}$. We then use Gaussian blur, for which the kernel looks as

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{7.16}$$

to remove the empty spots with and repeatedly dilate by taking an average in the given receptive field R simply as

$$I_O(x_o,y_o) = \frac{1}{N}\sum_{(x,y)\in R} I_I(x,y) \tag{7.17}$$

and thresholding against 0.5. This process generates a mask which we pass to the unsupervised loss function. Resulting segmentation looks as shown in figure 7.4.

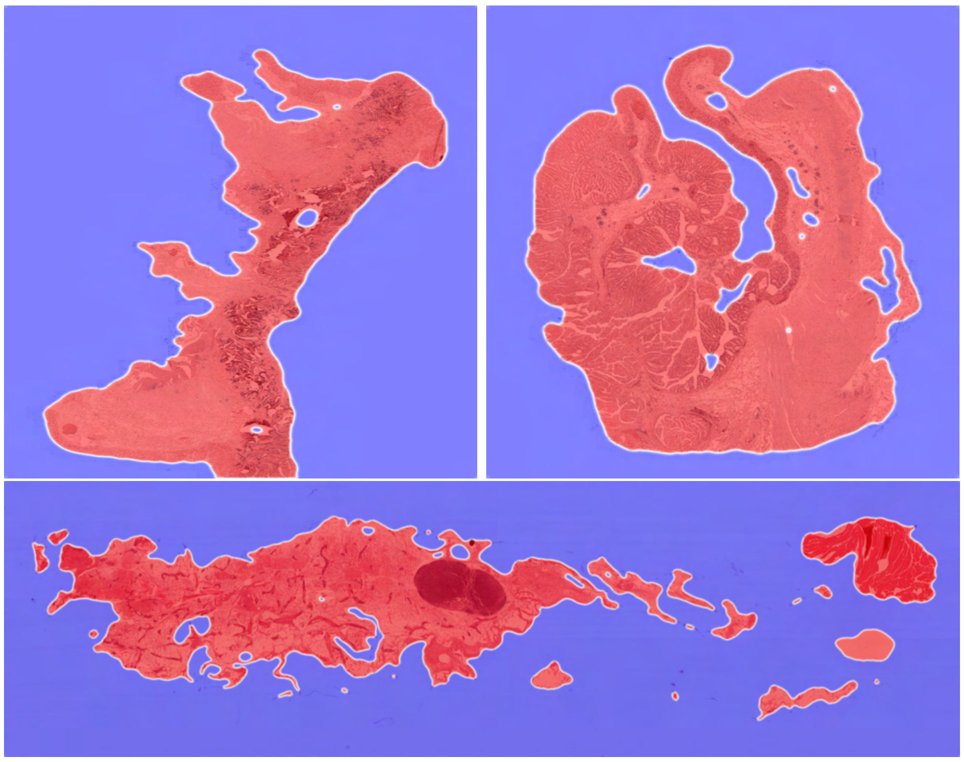


Figure 7.4: Foreground masks generated from our segmentation (red - foreground, blue - background)

Even though the method proposed above is relatively trivial, the results are acceptable. We do not need the mask to fit precisely onto the foreground, as this means slightly slower training without any actual loss of accuracy.

7.5 Image warping

The warping of the original image with the synthetic flow is done by transforming the pixel grid of the original image with the values of the flow as

$$C_O(\mathbf{x}) = C_I(\mathbf{x}) + \mathbf{f}(\mathbf{x}) \tag{7.18}$$

and the use of bilinear sampling in the coordinates $C_O(\mathbf{x})$ to transform the original image.

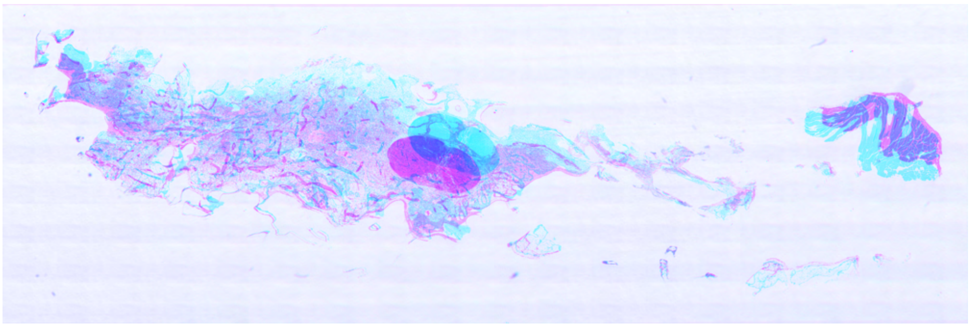


Figure 7.5: Example of a warped image (red - original, blue - warped)

7.6 Histogram matching

Histogram matching (histogram specification) is a process where the histogram of the original image is matched with the histogram of the target image. This is done by calculating the probability density function (PDF) as

$$p(x_i) = \frac{n_i}{N} \tag{7.19}$$

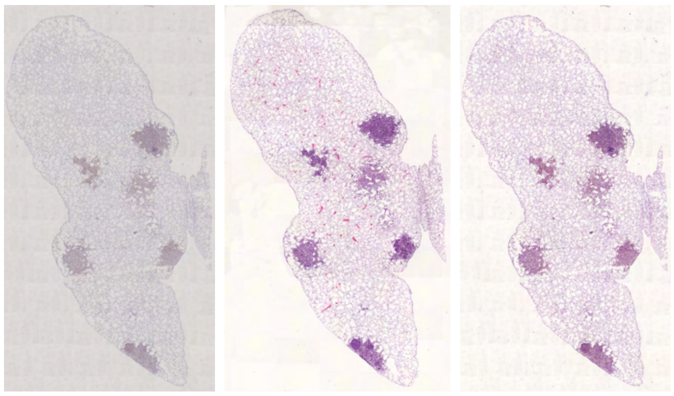
where x_i is the given color value, n_i is the number of pixels with this color, and N is the total number of pixels in the image. For scalar images, cumulative density function (CDF), is calculated for each of the images as

$$F_{(\cdot)}(x_i) = \sum_{j=1}^{i} p(x_j), \tag{7.20}$$

$i \in [0; L]$, where L is the number of different values, each color channel can have. Output value for each color is then

$$y_i = H^{-1}(F(x_i)), \tag{7.21}$$

where H is the CDF of the target image, F the CDF of the original image and x_i is the original value. [65]



(a) : Original image **(b)** : Target image **(c)** : Matched image

Figure 7.6: Histogram matching process

7.7 Simulated training dataset

For the actual training of our network, we use training triplets from the methods stated above. The dataset contains 2000 images for training and 200 images for validation. Images are interpolated from the original dimension of several thousand pixels to a more manageable size of $512 \times S$, where 512 is always the smaller side, and S is chosen so that the aspect ratio of the original image remains the same ($S \times 512$ size is used when the original image was wider than taller).

The transformation in this dataset is artificial (same for the target image) since we want to have an exact flow. The pipeline for the creation of the triplets is

1. Matching histograms of the original image to the target image to simulate the change in color, which is present in the ANHIR dataset.
2. Generate random flow as described in section 7.3. The maximum displacement of this flow is roughly tens of pixels (depending on the image size).
3. Random affine transformation is generated from the matrix 7.5. While we fix the parameter $a_{21} \approx -a_{12}$ in order to simulate rotation. Parameters a_{13} and a_{23}, which are responsible for the translation in the x and y directions, respectively, are scaled by the image dimensions.
4. Both the flows are added together, and the original image is warped to create the target image.

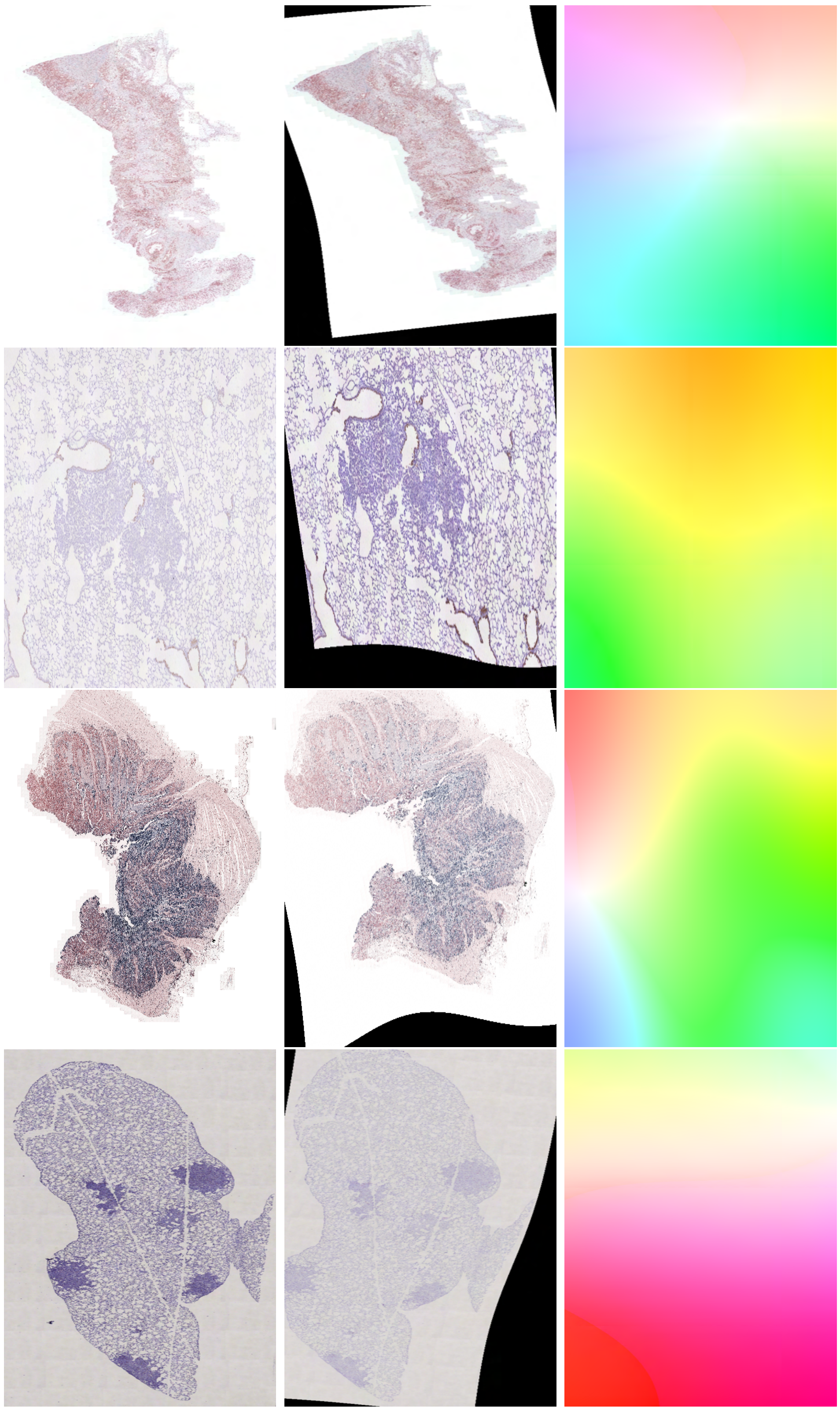


Figure 7.7: Examples of the training triplets generated from the combination of the affine and non-rigid transformation

Chapter 8

Training

The main goal is to train our network based on the RAFT architecture (either RAFT itself or the GMA) [57, 64] to become able to register images in the ANHIR dataset. All training was done with the AdamW optimizer and either fully supervised with a sequential loss function (see section B.5), a combination of this function and several unsupervised ones based on image similarity functions (see section 4.1). Gradients are clipped to the $[-1, 1]$ range for stability. Training took place on several GPUs based on availability, but mainly on the NVIDIA GTX 1050 Ti, with an SSD and Intel i7-8750H CPU.

8.1 Training on the synthetic datasets

The training process can be split into several different phases. First, the network is trained on the three synthetic datasets, Flying Chairs, FlyingThings 3D (similar to the Flying Chairs), and MPI-Sintel. Movement in these datasets is generally more straightforward and the magnitude smaller, usually locally rigid. While not having any realistic movements, the first two datasets are way simpler in appearance. MPI-Sintel is created from a short animated movie. The movement is therefore based on the movement in the real world. Parameters for this part of the training process are shown in the table 8.1.

Name	# of iterations	Batch size	Patch size
Flying Chairs	120k	6	368×496
FlyingThings3D	120k	6	400×720
MPI-Sintel	50k	6	368×768

Table 8.1: Parameters of the synthetic datasets

We use pre-trained weights, which are publicly available from the RAFT paper [57], since training on our machine with such a large batch and image size is not possible on our machine, and the reduction in any of these parameters would only lead to a reduction in the overall accuracy, with the required time to complete the number of iterations, being in the order weeks.

8.2 Training on the ANHIR dataset

We then switch to the ANHIR dataset, which creation was described in the chapter Simulated ANHIR dataset. Movement in the synthetic datasets has a maximum magnitude of a few hundred pixels. In contrast, movement in the ANHIR dataset sometimes has a magnitude close to a thousand pixels (even in our heavily down-scaled dataset). We, therefore, train a separate network responsible for global registration on lower resolution images.

We split the training on the ANHIR dataset into several different parts. First, we only use the ground-truth optical flow, which we generated (**Syn**) or the mixture of flow generated from affine transformation and directly from landmarks (**S**).

We introduce flow smoothness, SSIM, and census loss functions (see section 4.1) in the second part of the training. We combine the loss from these functions with the supervised loss, resulting in a semi-supervised learning (**SS**). Images used are partly from the dataset with the synthetic flow and partly from the one with approximate flow calculated between the real image pairs.

In the last part of the training, we switch entirely to unsupervised learning (**US**) and use image pairs exclusively from the ANHIR dataset for the fine-tuning. This is done under a very low learning rate of 10^{-5}.

Time per 1000 iterations is roughly **30** minutes, with the entire training requiring about **36** hours to complete.

Name	# of iterations	Batch size	Patch size
Supervised	40k	2	384x384
Semi-supervised	15k	2	384x384
Unsupervised	5k	1	512x512

Table 8.2: Parameters for the different parts of training on the ANHIR-based dataset

8.2.1 Training validation

A small validation dataset of 50 image pairs is randomly sampled from the ANHIR dataset. The AMrTRE (see section Evaluation metrics) of the pre-trained network is **0.0472** (see section 8.1), with the validation dataset having an initial value of **0.0469**. However, starting from the pre-trained model sped up the initial training drastically compared to starting from a blank model.

We see a sharp decrease in AMrTRE at the start of the training, with steady progress through the first 40 thousand generations. Once we switch to semi-supervised learning, we see a quick drop. The lowest error of **0.00762** is reached in the unsupervised learning in iteration 4k (59k iterations from the start of the training on the ANHIR-based datasets).

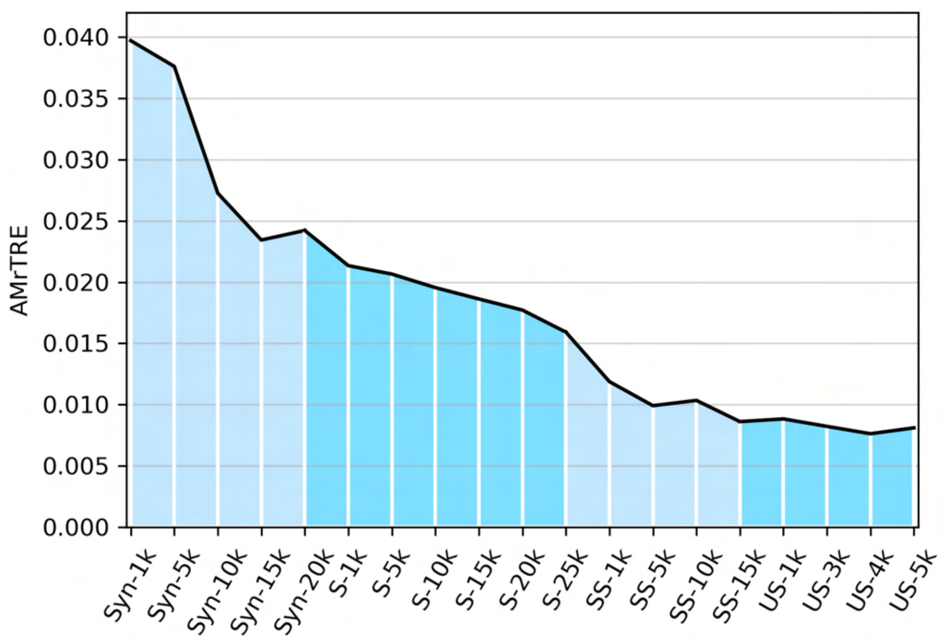


Figure 8.1: Validation dataset AMrTRE, (Syn - synthetic flow, S - supervised, SS - semi-supervised, UN - unsupervised; $(\cdot)k$ - thousands of iterations in the given stage)

8.3 Learning rate scheduler

For the learning rate scheduler, we use the one-cycle learning rate policy [66]. The learning rate is updated based on the number of iterations rather than the number of epochs. We start with a low learning rate, gradually increasing it to the predefined value, after which it is once again slowly lowered for the rest of the training [66].

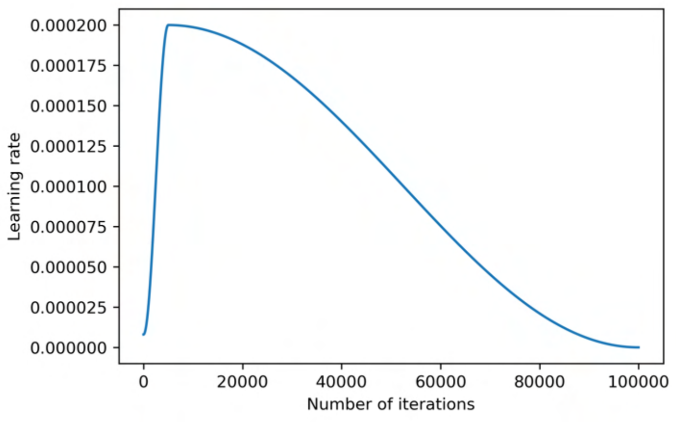


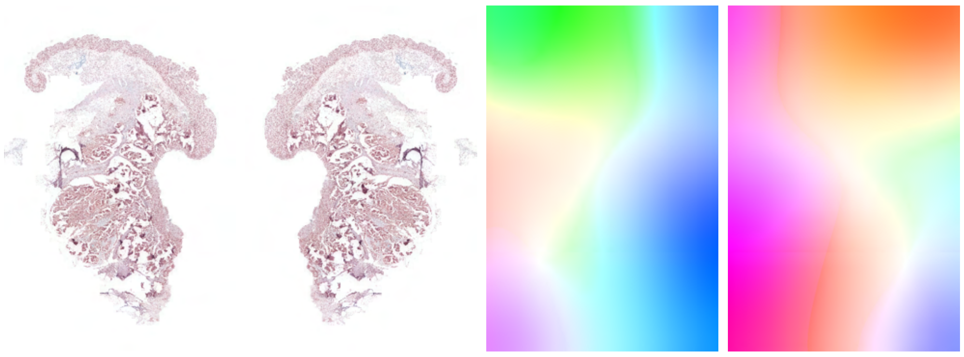
Figure 8.2: Learning rate progress under the one-cycle learning rate policy

8.4 Data augmentation

In order for the network to successfully register images in the ANHIR dataset, we implement several data augmentation techniques, which try to mimic different challenges we face in the ANHIR dataset. Another reason is to increase the effective amount of data we have available.

8.4.1 Flipping

One of the simplest yet effective data augmentation techniques is the random flipping of the images and the corresponding flow in either the vertical or the horizontal direction. After the flow is flipped, the direction of the flow needs to be reversed. This is done as $\mathbf{f}^{f}_{(\cdot)} = -\mathbf{f}^{n}_{(\cdot)}$, where f means flipped and n normal.

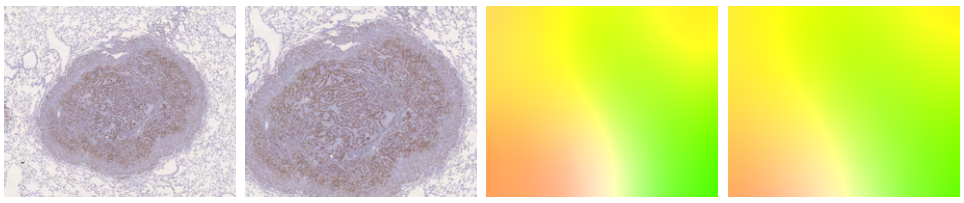


(a) : Original image **(b) :** Flipped image **(c) :** Original flow **(d) :** Flipped flow

Figure 8.3: Flip augmentation

8.4.2 Scaling

Image scaling means simulating different distances of the sample from the objective while the photo was taken while allowing the network to not fixate on features on a single scale. Images are interpolated to the new size and the flow is in addition multiplied with a scale factor as $\mathbf{f}^{r} = [x_{scale}, y_{scale}] \cdot \mathbf{f}^{n}$ because the magnitude of the movement is now different. Additionally, we can scale the dimension with different coefficients to stretch the image.



(a) : Original image **(b) :** Scaled image **(c) :** Original flow **(d) :** Scaled flow

Figure 8.4: Scale augmentation

8.4.3 Color augmentation

Aside from geometric differences, color change is the second main difficulty we face. Since image pairs in the ANHIR dataset are likely not to share the same color space, we simulate this with color augmentations (aside from histogram matching already done to the base images) by transforming the RGB images into the HSV and multiplying each channel by a random constant. This operation is asymmetric, meaning only one of the images undergoes this transformation.

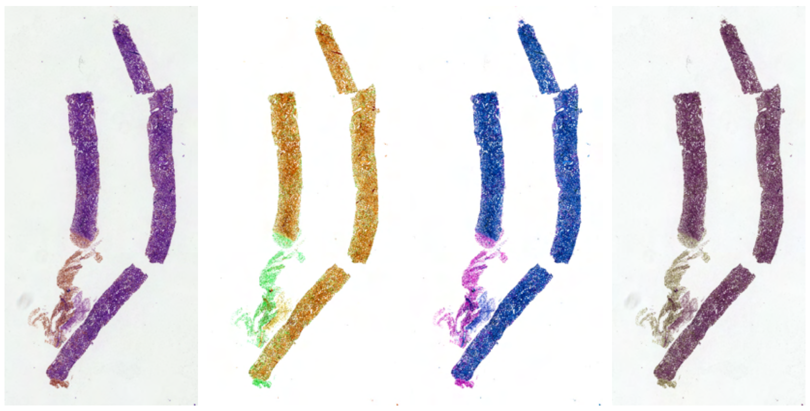


Figure 8.5: Examples of the color augmentations (first image is the original)

8.4.4 Additive noise

Further augmentation is done on a per-pixel basis. This is achieved by adding white Gaussian noise (AWGN) as

$$I_O(\mathbf{x}) = I_I(\mathbf{x}) + \mathcal{N}(0, \sigma^2), \tag{8.1}$$

where $\mathcal{N}$ is a random sample from the normal distribution with the mean of 0 and variance σ^2. Since the stains highlight different parts of the tissue, AWGN somewhat approximates this process.

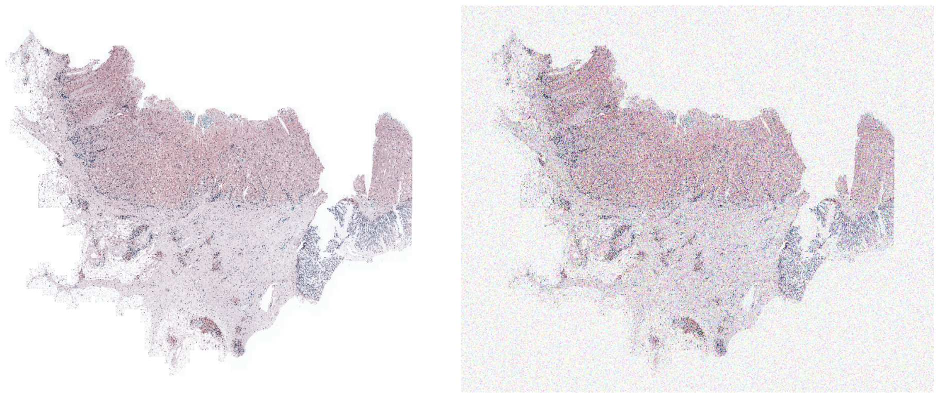


Figure 8.6: Additive white Gaussian noise (not to scale, higher value was used for the purpose of visualization)

8.4.5 Erasure

The tissue samples can change their shape drastically due to the folding or tearing, where entire areas of the image are missing. We randomly create several rectangles on the image and fill them with the mean value of the image.

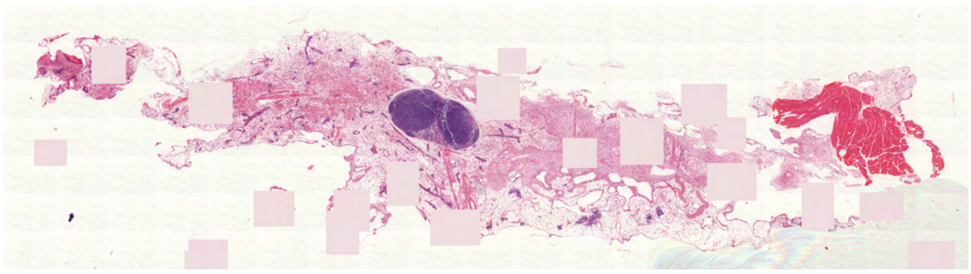


Figure 8.7: Erasure augmentation

8.5 Unsupervised loss functions

Several unsupervised loss functions are used in the second and third stages of the training process. Namely, these are SSIM, census loss (see section 4.1), and flow smoothness. Flow smoothness loss L_s is implemented as

$$L_s(\mathbf{f}) = \frac{1}{N} \sum_{\mathbf{x} \in \mathbf{f}} \nabla \mathbf{f}(\mathbf{x}) \tag{8.2}$$

We also utilize a multi-resolution approach when creating the SSIM and census loss, where instead of taking the image at a single resolution, we calculate the loss of an image pyramid. Combined unsupervised loss is then

$$L(I_1, I_2, \mathbf{f}) = w_s L_s(\mathbf{f}) + \sum_{l \,\in\, (C, SSIM)} w_l \sum_{i=1}^{N} \gamma^{N-i-1} L_l(I_1^i, I_2^i), \tag{8.3}$$

where $I_{(\cdot)}^i \in \mathbb{R}^{H/2^i \times W/2^i}$ are the downsampled grayscale images, $\mathbf{f}$ is the flow, γ is the exponential weight (0.8), and N is the number of levels of the image pyramid.

Since we need the warping to be differentiable, we use a spatial transformer module, introduced in Spatial transformer networks [67].

The census loss implemented differs from the standard definition. While normally, only integer values generated from thresholding are allowed, the approach which we use creates a list of real numbers. As such, the loss function is differentiable.

The weights of the individual loss functions $w_{(\cdot)}$ are not constant but are generated from a combination of sigmoids, which take the number of iterations as their input.

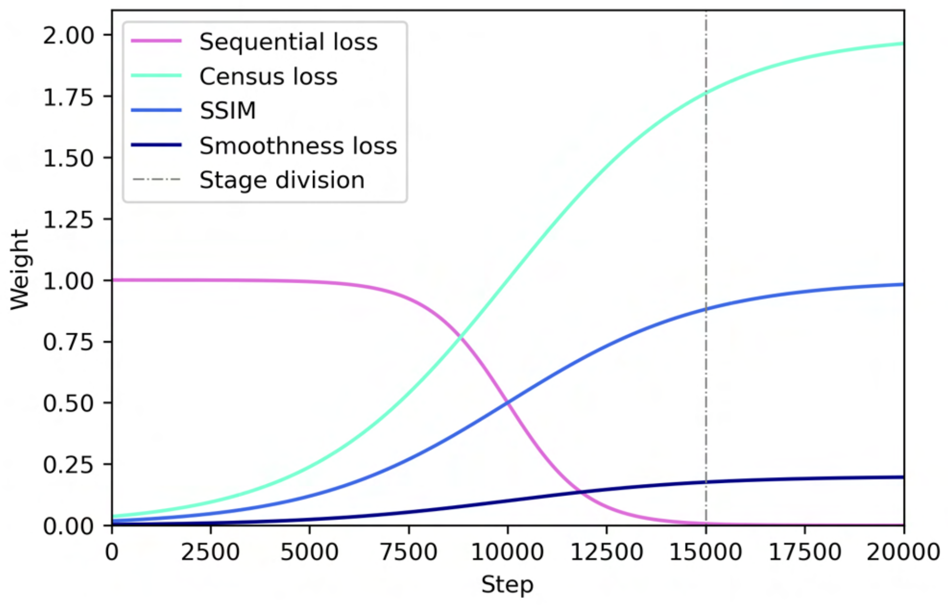


Figure 8.8: Weights of different loss functions (before additional scaling)

In the figure 8.8, we can see the semi-supervised (left) and unsupervised (right) parts of the training process. Even though the supervised weight is only 1, the sequential loss is bigger for most of the semi-supervised part than the unsupervised losses, with sequential loss reaching values of $5 - 100$ and unsupervised losses usually staying below 1. Since, in this stage, they are used together, and their values are comparatively not very high, we need to increase their weights to see any results from their usage. This is done simply a multiplying all three by 10. The value of 10 was chosen arbitrarily and, therefore, may not be optimal.

Chapter 9

Experiments and results

9.1 Landmark warping

In order to quantify the resulting accuracy of our network, a set of landmark pairs is utilized for each image. The landmarks from the first image need to be warped towards the landmarks in the second image, after which point, metrics listed in section 3.5 are calculated.

Since we cannot directly obtain the coordinates of the warped landmarks after flow warping, we need to find a point in the warped coordinate grid with the smallest distance to the original landmark coordinate, as this is the point from which we want to sample in order to get the second image.

The warped landmark is found as

$$l_W(i) = \underset{\mathbf{x} \in G_T}{\operatorname{argmin}} \|l_1(i) - \mathbf{x}\|_1, \tag{9.1}$$

where G_T is the warped coordinate grid of the original grid G, calculated as $G_T(\mathbf{x}) = G(\mathbf{x}) + \mathbf{f}(\mathbf{x})$ and $l_{(\cdot)}$ are the landmarks. In order to increase accuracy, we sample 4 closest points, which are then weighted as

$$l_W(i) = \frac{\sum\limits_{i \in \text{4-NN}} (d_i)^{-1} \mathbf{x}_i}{\sum\limits_{i \in \text{4-NN}} (d_i)^{-1}}, \tag{9.2}$$

where d_i is the distance of the point $\mathbf{x}_i$ in the coordinate grid.

9.2 Ablations

A set of ablation experiments is performed to show the importance of each of the components of our method. The experiments were performed in isolation, sometimes on differing sub-samples of the dataset (method structure and similarity check was run on the full dataset). Results of each study is shown in table 9.1. The setting which we will use in our method is underlined. Each of the experiments is described in more depth below.

Experiment	Setting	AMrTRE	AMxrTRE	AR
Patch size	96	0.038214	-	-
	128	0.018328	-	-
	256	0.004538	-	-
	384	0.002421	-	-
	512	0.002343	-	-
Patch frequency	1	0.002493	-	-
	2	0.002401	-	-
	4	0.002353	-	-
	7	0.002316	-	-
	10	0.002323	-	-
Flow updates	1	0.010071	-	-
	2	0.009164	-	-
	5	0.009136	-	-
	7	0.089402	-	-
	9	0.009091	-	-
	12	0.009370	-	-
Method structure	Global-only	0.02292	0.05367	0.9060
	Local-only	0.01275	0.03945	0.9656
	Course-to-fine	0.01005	0.03609	0.9579
Similarity check	No	0.01005	0.03609	0.9579
	Yes	0.00730	0.03065	0.9790

Table 9.1: Ablation settings

9.3 Patch based registration

The local module splits the original image into several smaller overlapping patches. This process is controlled by two parameters, namely the size of the patch and the frequency of the overlap.

9.3.1 Weighted flow mask

Features are more likely to be registered correctly if they are in the middle of the patch, as the chance that the corresponding features are inside is higher than those at the edge of the patch. The mask aims to give flow estimates at the edge of the patch with lower weight than those in the middle in hopes that other patches will be centered in such an area. A weighted mask is created from a Gaussian function, which gives a higher value to the flow estimated in the middle, as shown in the figure 9.2.

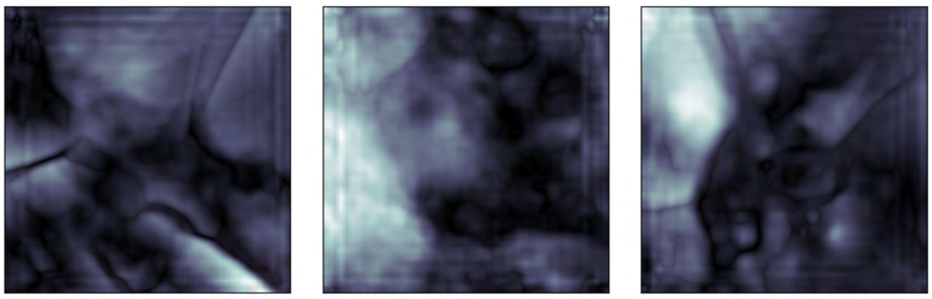


Figure 9.1: Flow prediction error (scaled to the 0-1 range, where white corresponds to the maximum error within the given patch)

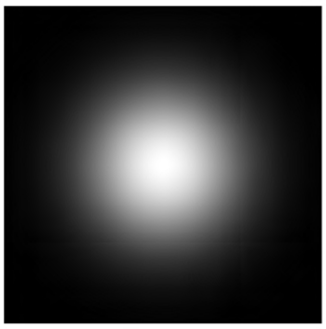
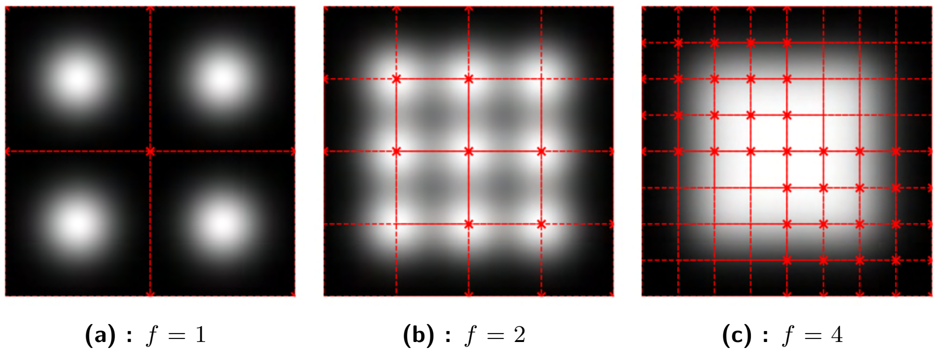


Figure 9.2: Patch weight mask (white - 1, black - 0)

The image is registered with overlaying windows, with varying frequency, with higher density in the middle, as most of the tissue samples are already centered and do not touch edges. Weight masks are shown in the figure 9.3.



(a) : $f = 1$ **(b) :** $f = 2$ **(c) :** $f = 4$

Figure 9.3: Flow weights with varying frequency (shown are the edges of the patches and the top-left and bottom-right coordinates)

There are several parameters that we have chosen arbitrarily. Namely, these are the patch size, frequency, and variance of the Gaussian function. Variance has been chosen for each patch size individually so that the mean value of weights inside the patch is normalized. To find optimal values, we run multiple trials for each of these parameters on a smaller subset of 50 image pairs with dimensions of $512 \times S$.

9.3.2 Patch size

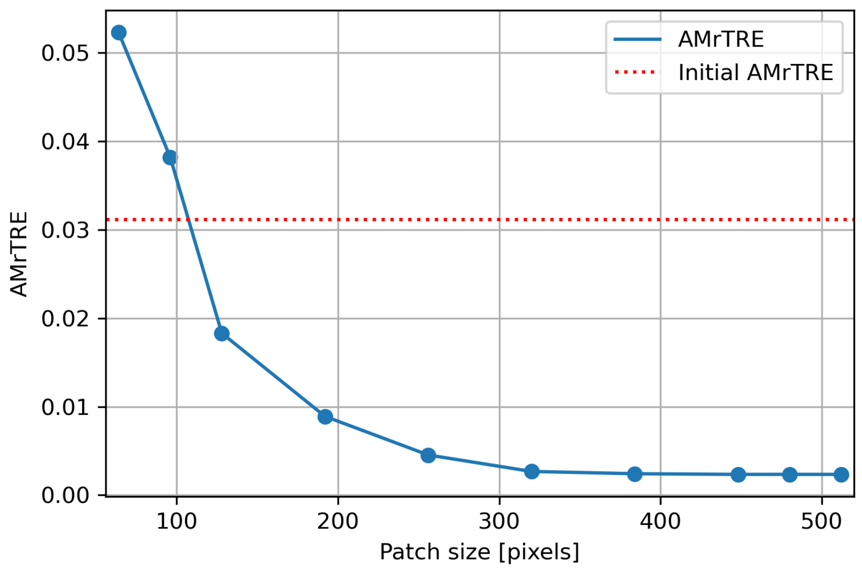


Figure 9.4: AMrTRE as a function of patch size

As we can see in the figure 9.4, the registration error decreases with the size of the patch. Below a certain threshold (around 110 pixels), the registration increases the distance of the landmarks, resulting in a complete failure. Once the patch gets big enough (around 320 pixels), any further increase in accuracy, while still measurable, is greatly diminished. However, a bigger patch size decreases the time of registration since a big part of the time is not spent on the actual registration, and doing so reduces this overhead. Therefore, we will always choose the biggest patch size, which the memory or the image size allows. Measurements were done with the frequency value of 4.

9.3.3 Frequency

We will test several different frequencies, namely $f \in [1, 10]$. To increase the number of times the given point is registered, the initial coordinates are shifted by a distance proportional to the patch size and frequency in order to offset them and allow the network access to a different context, which could have previously not been observable.

From our experiments depicted in the figure 9.5, we can see that the frequency above 6-7 does not improve the accuracy of the registration any further while increasing the time required drastically. Frequency of **7** will therefore be used in the future. Tests were done with a patch size of 512.

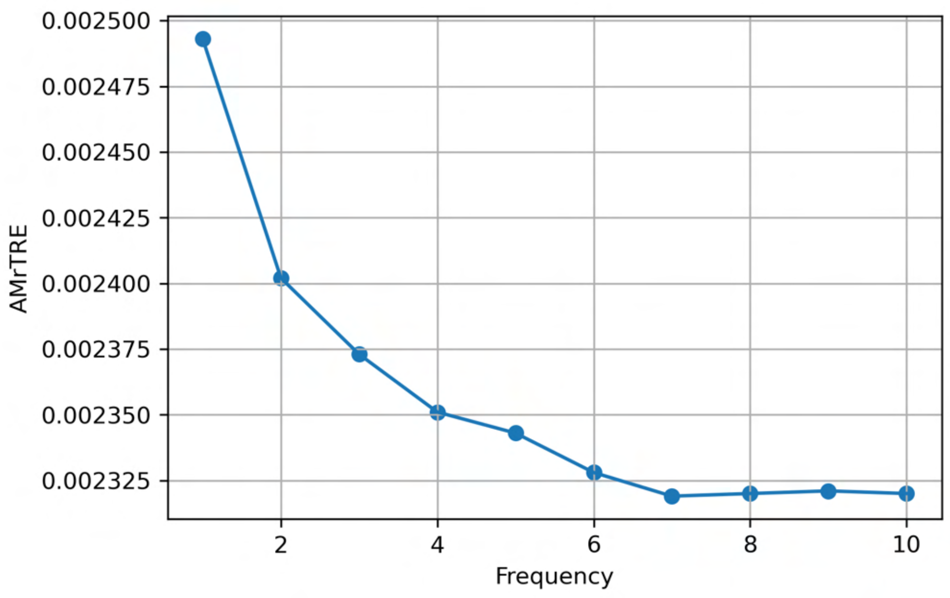


Figure 9.5: AMrTRE as a function of frequency

9.4 Effect of number of iterative flow updates on registration accuracy

Both the RAFT and GMA architectures update the flow estimate iteratively. The first several updates are responsible for the biggest displacements, with subsequent updates correcting the error done by the previous ones.

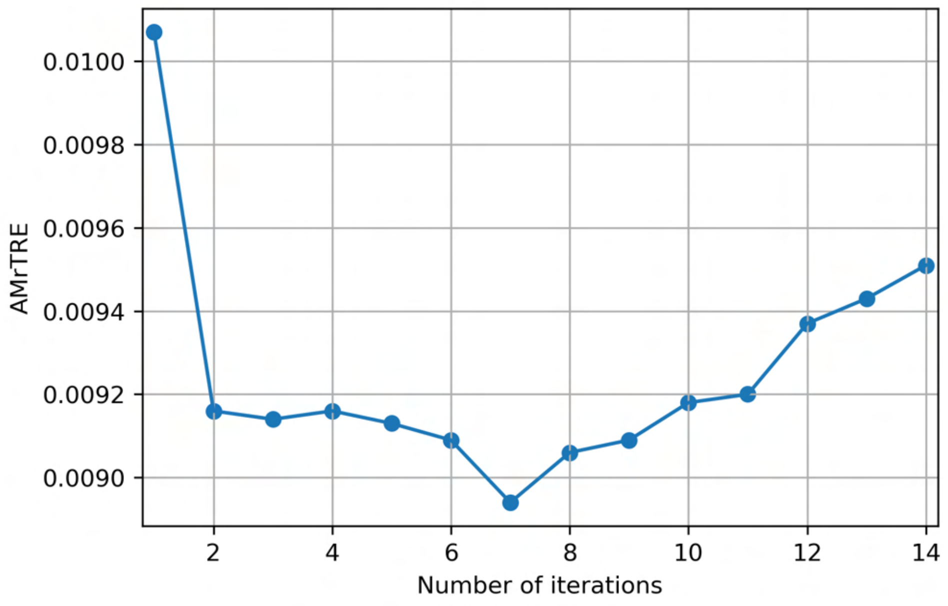
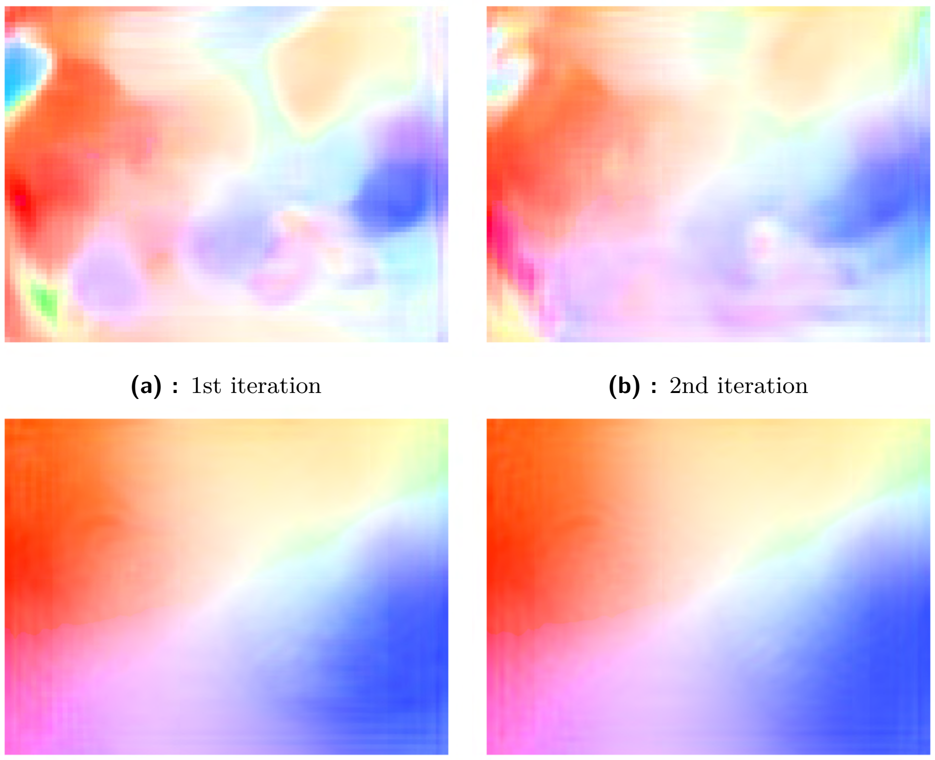


Figure 9.6: AMrTRE as a function of the number of iterations

The initial AMrTRE of the dataset sample was 0.04125. As we can see in figure 9.6, the most significant decreases in AMrTRE occurs in the first update, where the value gets lowered to roughly 25% of the original. The second update reduces the error further by 10% of the previous value. We then see a slight decrease until we reach a minimum in the **7th** iteration, after which point, the error starts increasing.

This is in contrast to the findings of the original RAFT paper [57], which says that an increase in the number of residual flow updates does not reduce the accuracy, but the amount reduced becomes insignificant after 12 updates.

One of the possible causes for the diverging accuracy might be that once a certain flow precision is reached, any further updates only cause the flow to become smoother and not correctly predict small local movements.



(a) : 1st iteration

(b) : 2nd iteration

(c) : 7th iteration

(d) : 12th iteration

Figure 9.7: Flow after increasing number of updates

In the figure 9.7, we can see flow prediction after 1, 2, 7 and 12 iterations. The flow progressively becomes more accurate, however between the 7th and 12th iteration, a loss of detail start to become visible.

9.5 Effectiveness of the global module

Since the tissue samples in the ANHIR dataset are somewhat centered and, except for a few cases, are entirely inside, we can be somewhat generous with the size of the images used for the global registration. Most of the images will have a flow magnitude of roughly 13.5% of the original image size, with outliers reaching 23.8% [9]. Images are downsampled to roughly $250 \times S$ size, depending on the original image size. This size allows for the global view of the image while keeping the resolution at an acceptable level. Lower values cause a loss of visibility in important structures and mainly cause edges of the tissue to become blurred.

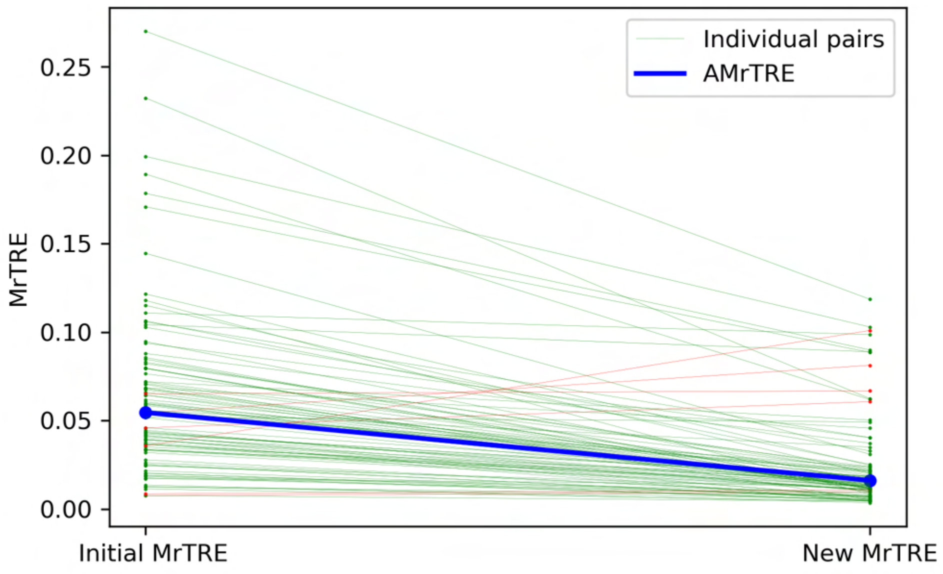


Figure 9.8: Performance of the global module (green - reduction in MrTRE, red - increase in MrTRE, blue - AMrTRE)

As we can see in the figure 9.8, the proposed global module is capable of reducing the registration error, especially in images with high levels of displacement. However, this reduction is not as visible for the smaller ones, as we need higher resolution to achieve better performance. The global network creates a smooth flow, which does not create artifacts in the warped image, and the said image is therefore usable in further registration. In our randomly sampled dataset, it was capable of reducing the AMrTRE to roughly **30**% of the original value.

The global network is purposely not capable of registering details. Such actions could lead to an increased error and further harm the chances for successful registration in the patch-based part.

9.6 Effectiveness of the local module

For the local module, we use settings found experimentally, namely patch size of 512 and patch frequency of 7. Furthermore, limiting the number of flow updates to 7 is especially beneficial. While the smoother flow estimate does not reduce the overall effectiveness of the global module, we need the flow to retain the finer details.

From the figure 9.9 (same sampled dataset as the one used for the global network), we can see that the local network performs better on images that already have somewhat low AMrTRE while struggling with the ones which do not. However, compared to the global network, it does not increase the registration error for any of the pairs.

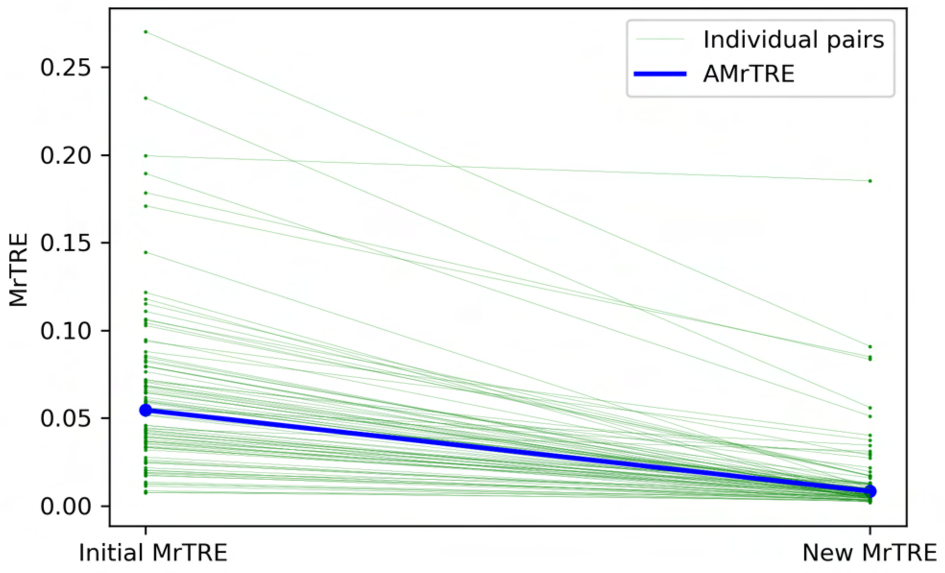


Figure 9.9: Performance of the local module (green - reduction in MrTRE, red - increase in MrTRE, blue - AMrTRE)

9.7 Similarity check

We compare census loss of the image pair before and after registration to decide whether the registration was successful and only apply those for which the ratio of census loss, computed as

$$r = \frac{L_c(I_W, I_2)}{L_c(I_1, I_2)} \tag{9.3}$$

where $L_c((\cdot), (\cdot))$ is the census loss of warped image I_W and original and target images, is lower then 0.90 (value chosen arbitrarily, for the given dataset). This prevents us from applying unsuccessful registrations and those which could be handled more precisely by using only the local module.

9.8 Effectiveness of the course-to-fine approach

Through a combination of the global and local networks, with a similarity check in between, to get rid of unwanted registrations, our method can further reduce the registration error. Compared to the local-only approach, we are seeing a reduction of roughly **42.75**% and **68.1**% when compared to the global-only approach.

Out of the three possible registrations (global only, local only, course-to-fine), the course-to-fine approach achieves the best results in **73.90%** of the cases, followed by local only, standing at **24.36%** and global only at **1.74%**, with the multi-scale census loss being able to determine, whether the global registration was successful in almost all of the cases.

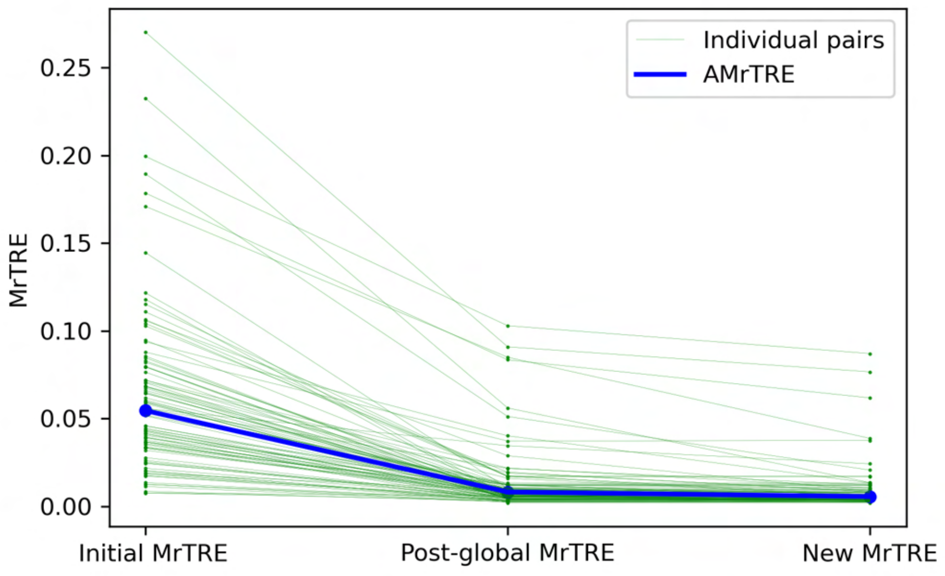


Figure 9.10: Performance of the course-to-fine approach (green - reduction in MrTRE, red - increase in MrTRE, blue - AMrTRE)

9.9 Error generated by the network

Unfortunately, the network generates a small amount of error not caused by the displacement of the tissue itself but the inability of the network to correlate the two images correctly. This amount is in a range that significantly decreases the final accuracy. We estimate this error by letting the network register a set of duplicate images. On the small dataset, landmarks have had their AMrTRE increased from zero to **0.0014**, with each landmark having been moved by approximately **1.5** pixels.

With the resulting AMrTRE being approximately 0.0073 (7.8 pixels), the noise values are equivalent to **19.2**% of the final error.

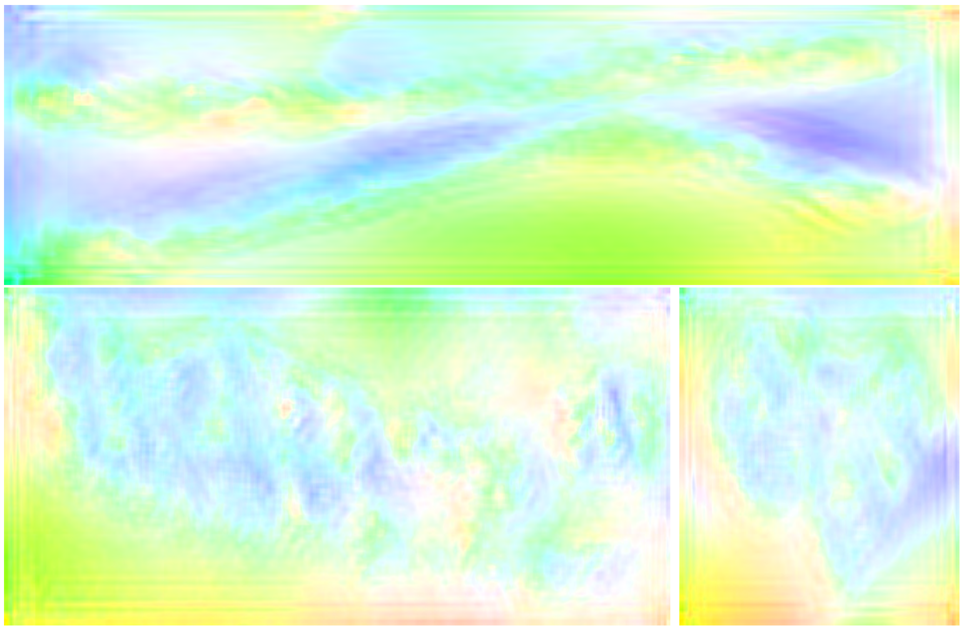


Figure 9.11: Error generated by the network

9.10 Results

In sections that follow, we will quantify the results of our method by calculating the metrics presented in section 3.5 and compare them with other participants of the ANHIR challenge.

Results were computed with our scripts rather than uploading our method to the ANHIR challenge.

9.11 Results of our method per sub-dataset

	ArTRE		MrTRE		MxrTRE		Robustness	
Name	avg	med	avg	med	avg	med	avg	med
Breast tissue	0.0102	0.0108	0.0091	0.0085	0.0312	0.0317	1.0000	1.0000
COAD	0.0099	0.0056	0.0082	0.0043	0.0373	0.0241	0.9806	1.0000
Gastric tissue	0.0046	0.0034	0.0041	0.0031	0.0132	0.0101	0.9989	1.0000
Human kidney	0.0036	0.0034	0.0030	0.0028	0.0115	0.0101	0.9969	1.0000
Lung lesion	0.0075	0.0075	0.0064	0.0062	0.0241	0.0231	0.9771	1.0000
Lung lobes	0.0040	0.0035	0.0032	0.0027	0.0198	0.0179	0.9857	0.9907
Mammary glands	0.0068	0.0063	0.0056	0.0047	0.0227	0.0184	0.9854	1.0000
Mice kidney	0.0233	0.0199	0.0200	0.0160	0.0635	0.0582	0.9242	0.9242
All	**0.0083**	**0.0056**	**0.0073**	**0.0045**	**0.0306**	**0.0211**	**0.9796**	**1.0000**

Table 9.2: Results of our method per sub-dataset (AMrTRE is in the avg MrTRE column)

The most important metric is the AMrTRE. Our method can reduce its value below 0.01 (representing 1% of the image diagonal) for every sub-dataset except the mice kidney. This sub-dataset also scores the worst in every other metric.

As we can see from the robustness metric, our network is capable of reducing the landmark distance for **97.66**% of the landmark pairs.

The time required for registration varies widely, depending on the image size, ranging from roughly 8 seconds on smaller images like lung lobes or lung lesions up to 45 seconds on larger images like mammary glands.

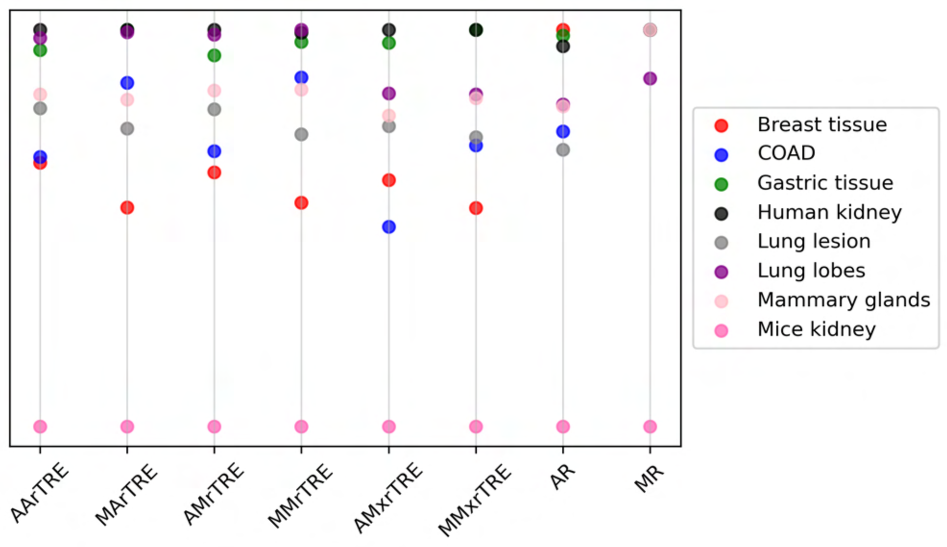


Figure 9.12: Relative results of our method per sub-dataset, with values normalized and scaled for visualization (higher is better)

9.12 Comparison to state-of-the-art

	ArTRE		MrTRE		MxrTRE		Robustness		
method	avg	med	avg	med	avg	med	avg	med	Time [min]
initial	0.1340	0.0684	0.1354	0.0665	0.2338	0.1157	-	-	-
MEVIS	0.0044	0.0027	0.0029	0.0018	0.0251	0.0188	0.9880	1.0000	0.17
AGH	0.0053	0.0032	0.0036	0.0019	0.0283	0.0225	0.9821	1.0000	6.55
UPENN	0.0042	0.0029	0.0029	0.0019	0.0239	0.0190	0.9898	1.0000	1.60
CKVST	0.0043	0.0032	0.0027	0.0023	0.0239	0.0189	0.9883	1.0000	7.80
TUB	0.0089	0.0029	0.0078	0.0021	0.0280	0.0178	0.9845	1.0000	0.02
TUNI	0.0064	0.0031	0.0048	0.0021	0.0287	0.0204	0.9823	1.0000	9.73
DROP	0.0861	0.0042	0.0867	0.0028	0.1644	0.0273	0.8825	0.9892	3.99
ANTs	0.0991	0.0072	0.0992	0.0058	0.1861	0.0351	0.7889	0.9714	48.24
RVSS	0.0472	0.0063	0.0448	0.0046	0.1048	0.0275	0.8155	0.9928	5.25
bUnwarpJ	0.1097	0.0290	0.1105	0.0260	0.1995	0.0727	0.7899	0.9310	10.57
Elastix	0.0964	0.0074	0.0956	0.0054	0.1857	0.0353	0.8477	0.9722	3.50
UA	0.0536	0.0100	0.0506	0.0082	0.1124	0.0353	0.8209	0.9853	1.70
NiftyReg	0.1120	0.0372	0.1136	0.0355	0.2010	0.0714	0.7427	0.8519	0.14
Our method	**0.0088**	**0.0056**	**0.0073**	**0.0045**	**0.0307**	**0.0211**	**0.9796**	**1.0000**	**0.23**

Table 9.3: Comparison of methods, data taken from [9] (AMrTRE is in the avg MrTRE column)

Our method is comparable to results achieved by other participants of the ANHIR challenge while outperforming the baseline methods provided by the organizers. While the accuracy is on the lower end of the participant spectrum, the time required is the third lowest.

The reduction from the initial AMrTRE value of **0.1354** to **0.0073** means that our method was capable of a reduction of **94.6**%. This result places our methods on the **5**th place based on AMrTRE.

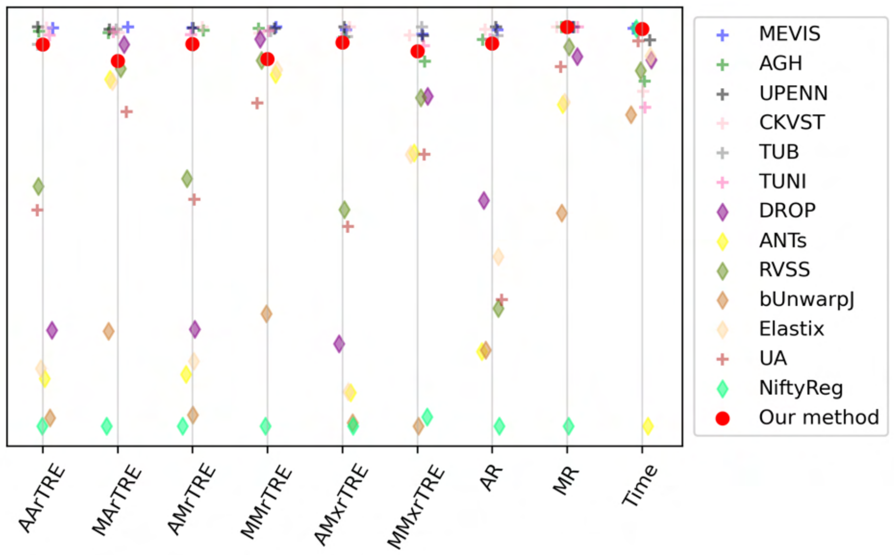


Figure 9.13: Comparison of methods, with values normalized and scaled for visualization (cross - participant method, diamond - organizer method, higher is better)

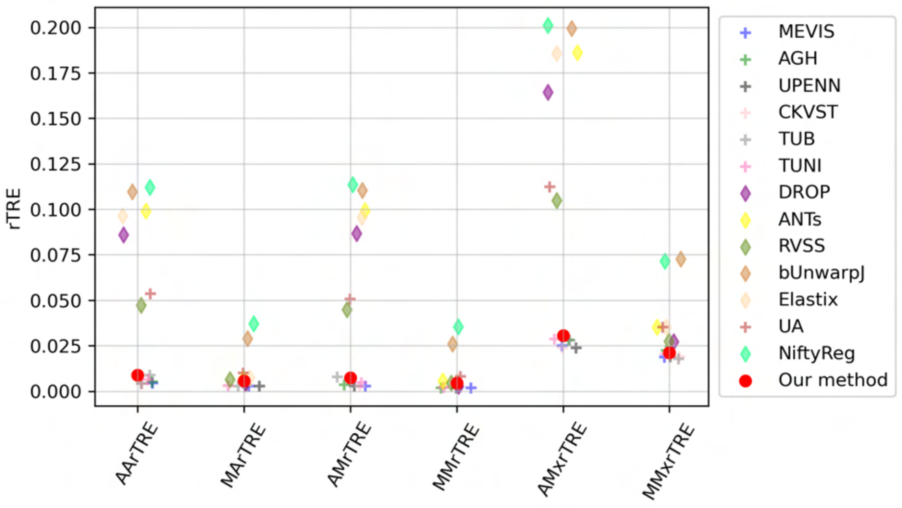
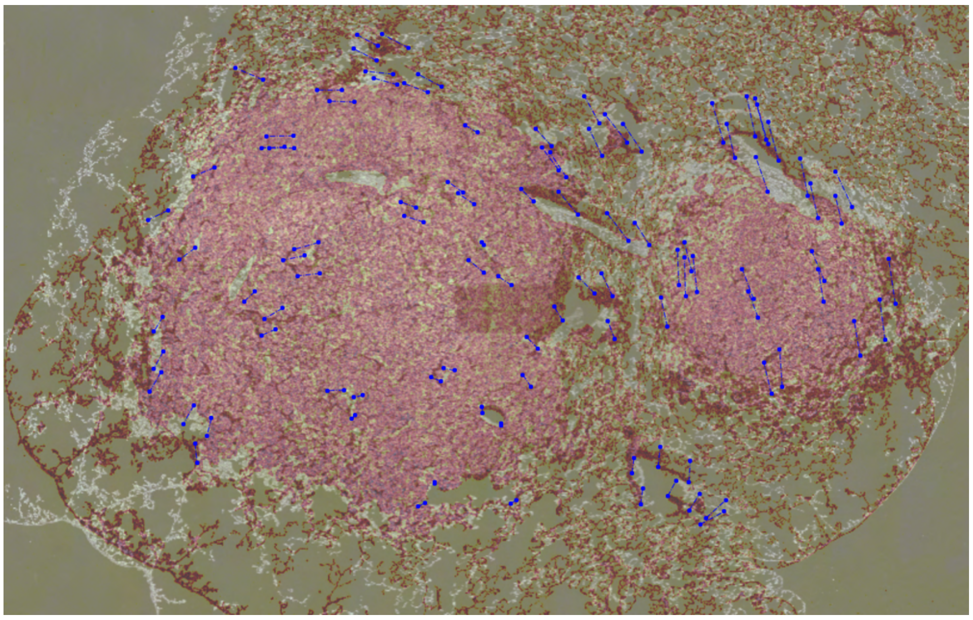


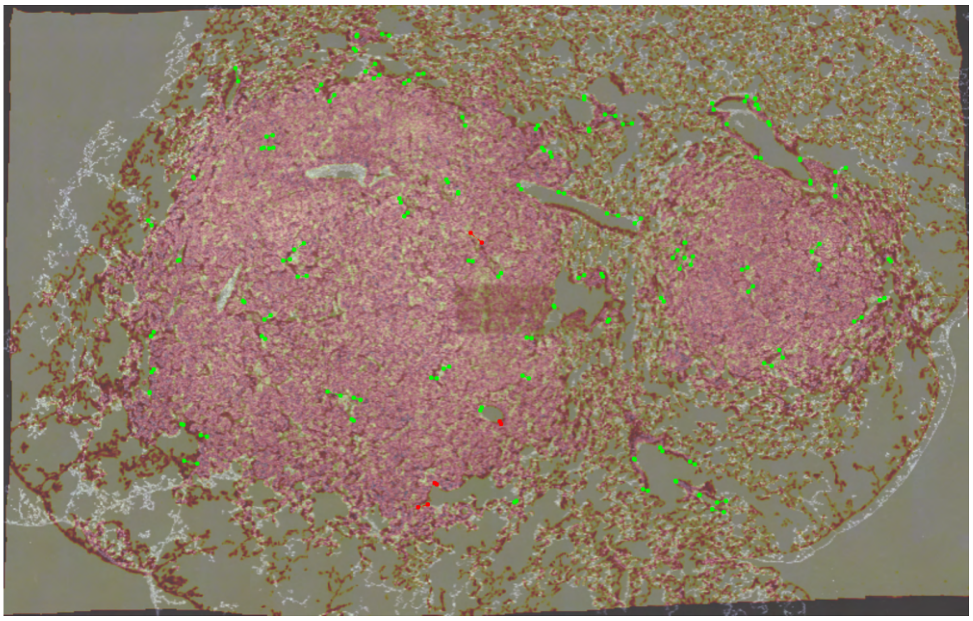
Figure 9.14: rTRE based metrics comparison (cross - participant method, diamond - organizer method, lower is better)

9.13 Examples of the registration

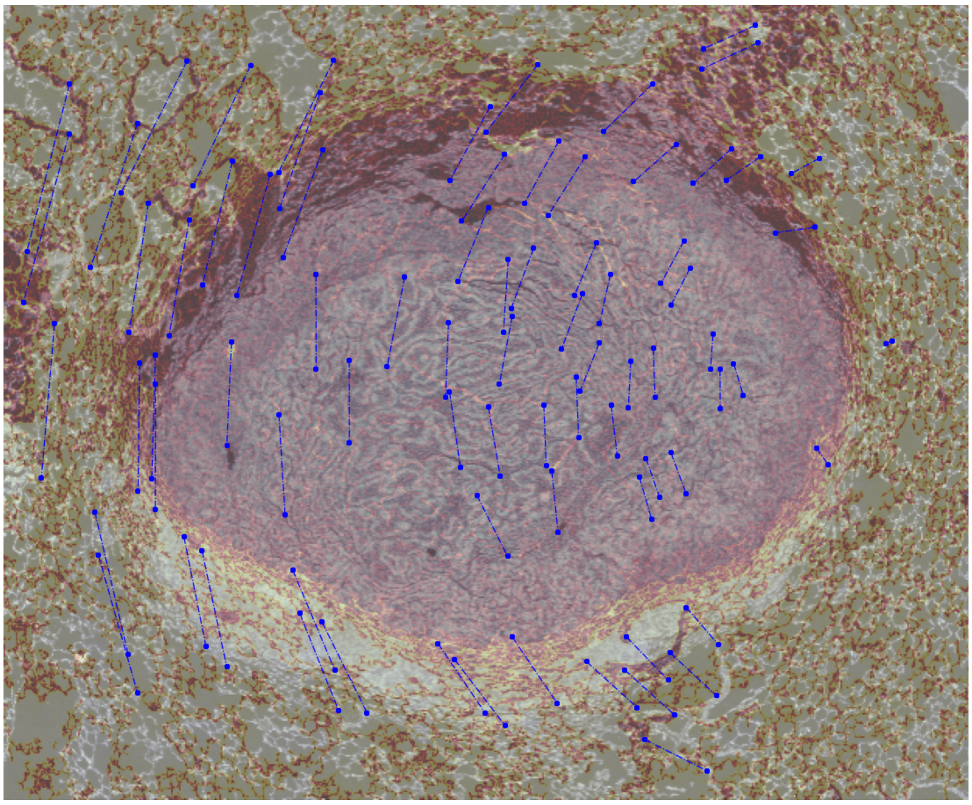
This section shows several image pairs which were registered using our method. The first image in each pair represents the original difference, with the respective landmark distance shown by blue lines. The second image shows the difference after registration, with landmarks that have reduced their distance shown in green and those which did reduce it in red.

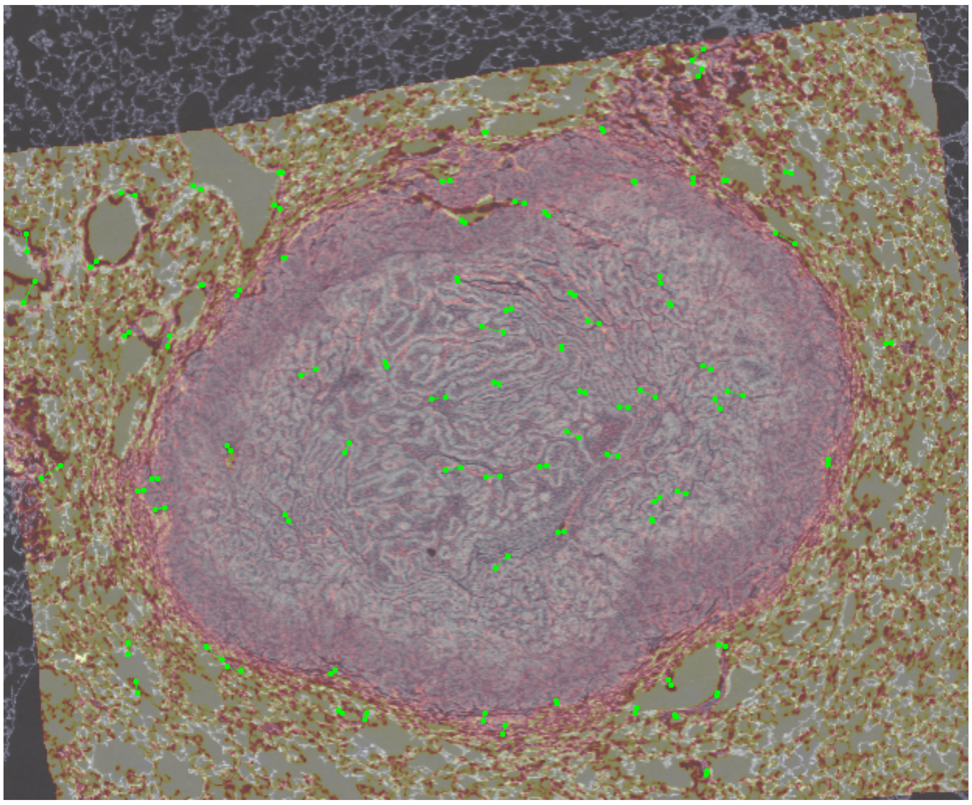


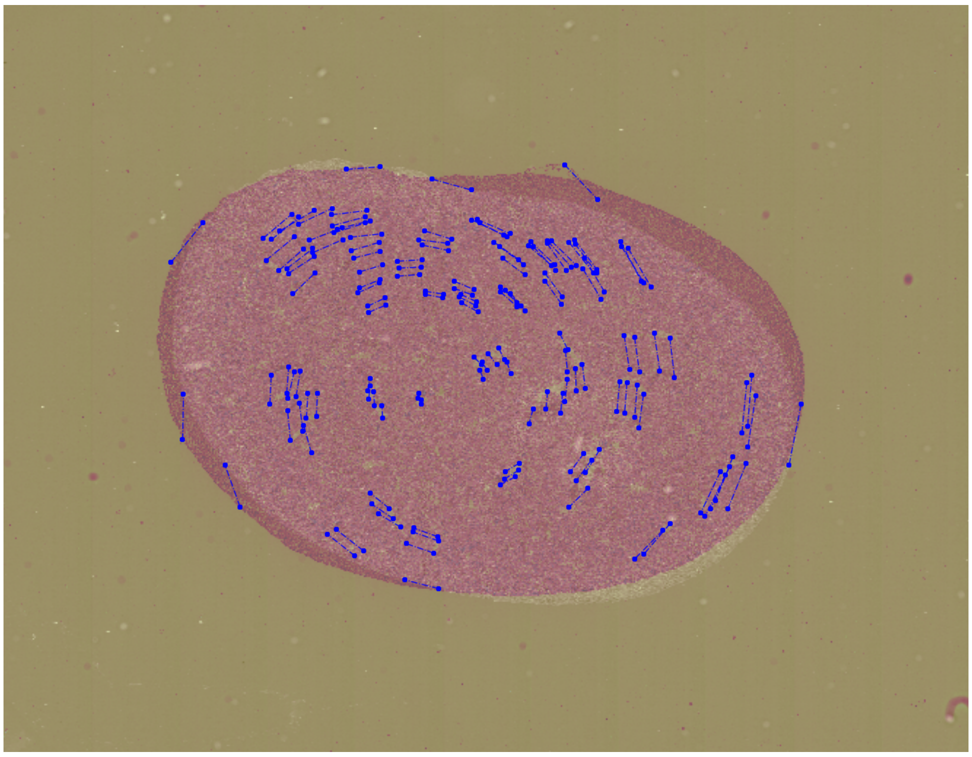
(a) : Original difference



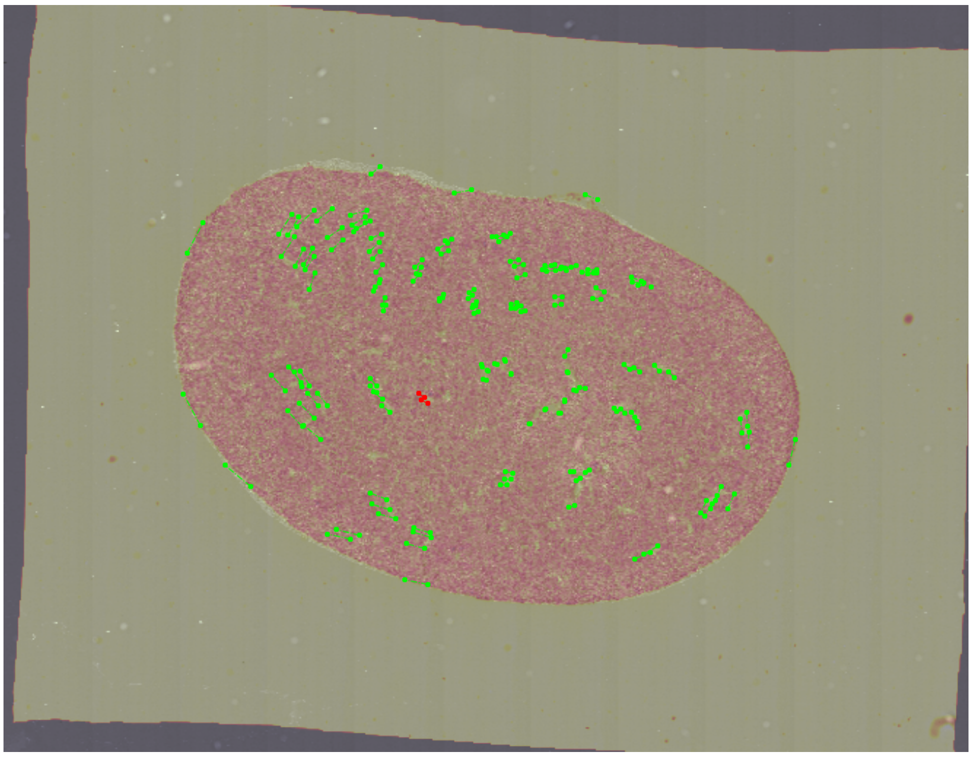
(b) : Warped difference


(a) : Original difference

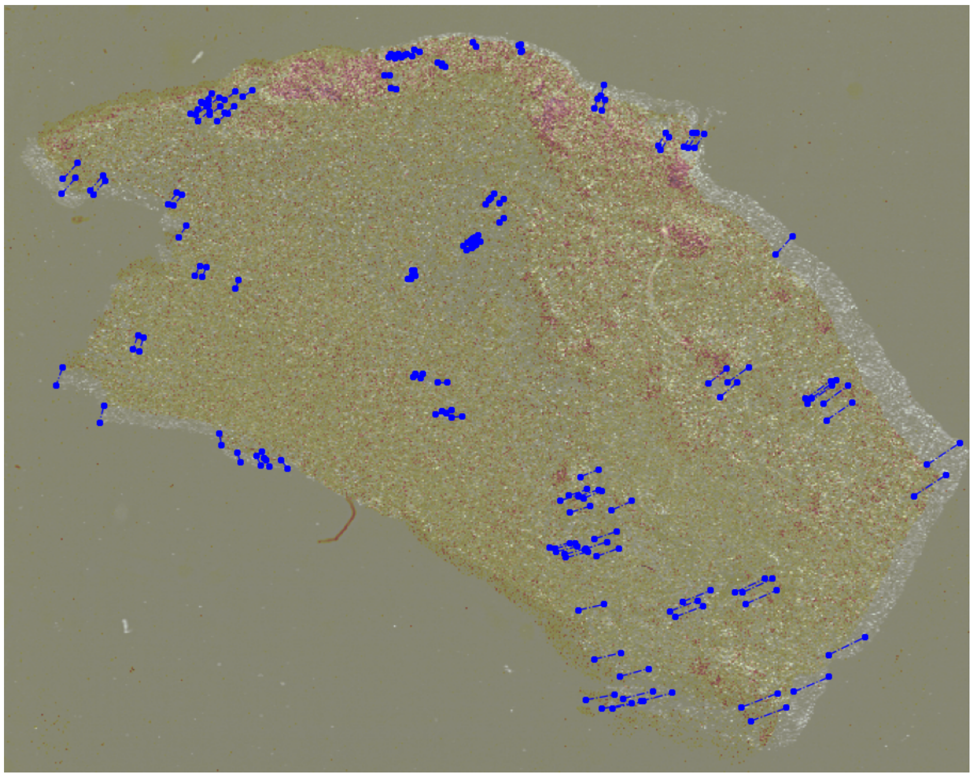

(b) : Warped difference

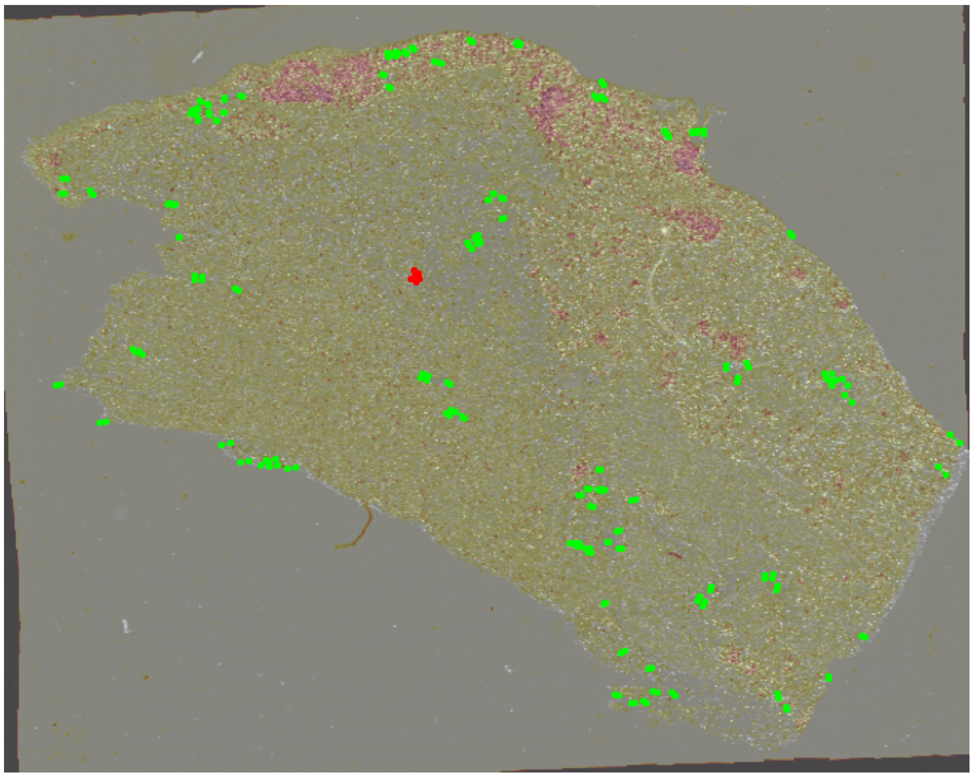


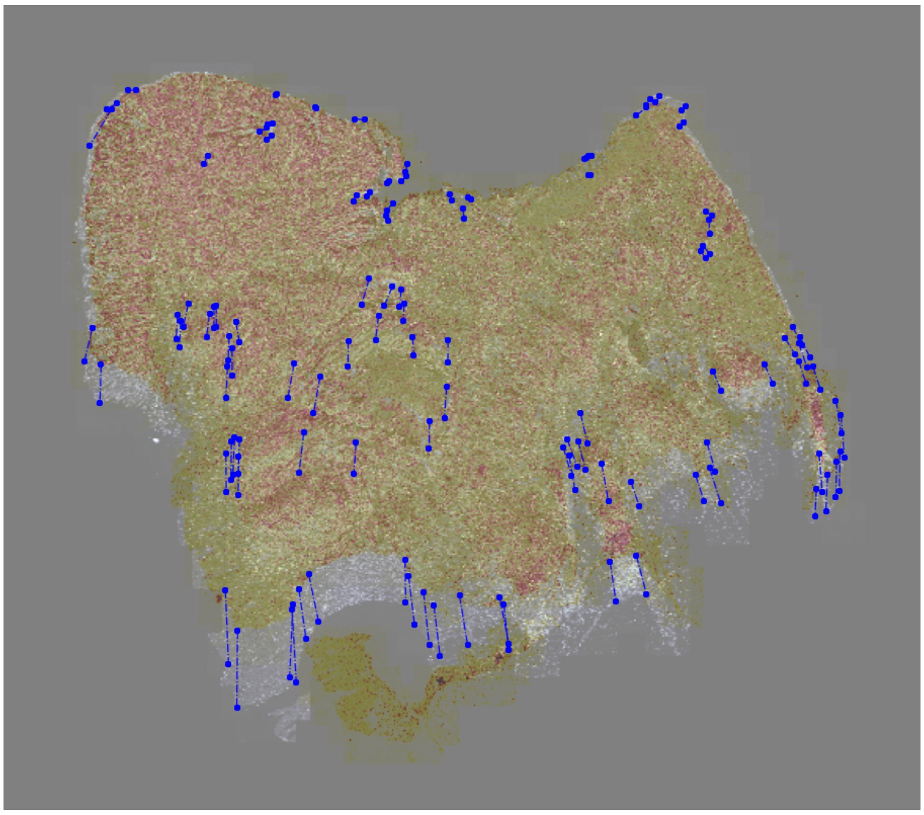
(a) : Original difference



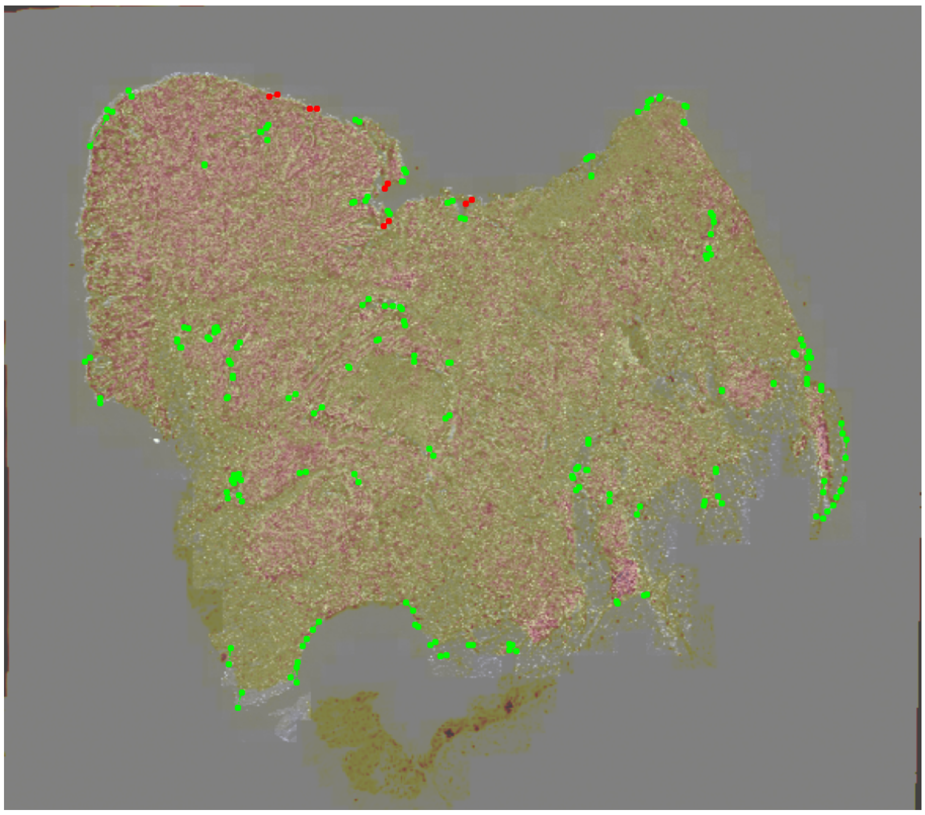
(b) : Warped difference


(a) : Original difference

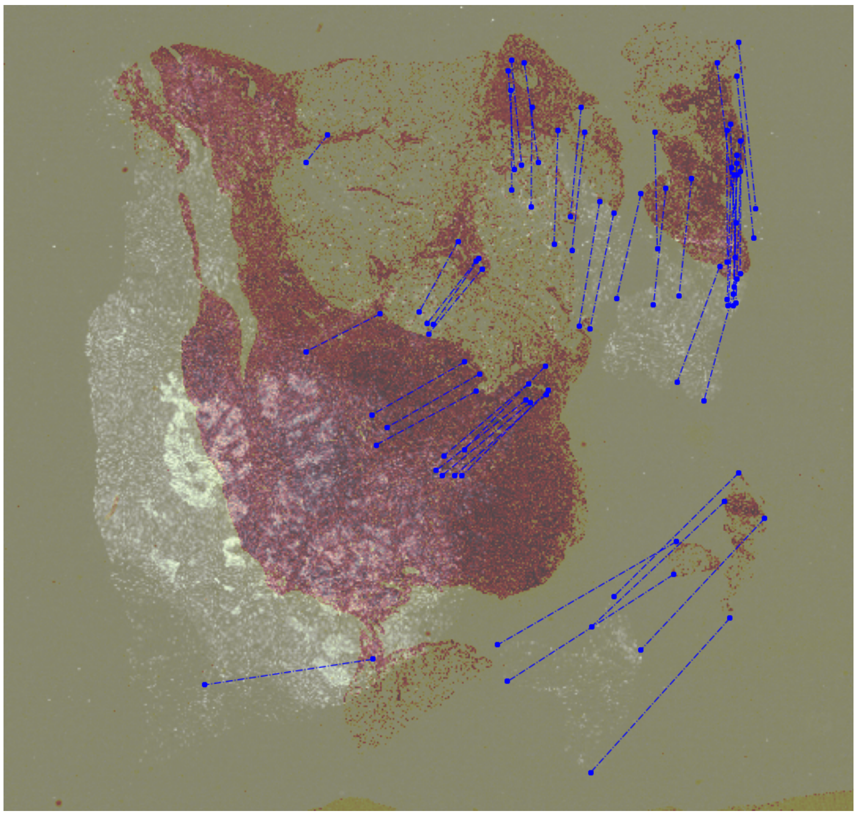

(b) : Warped difference



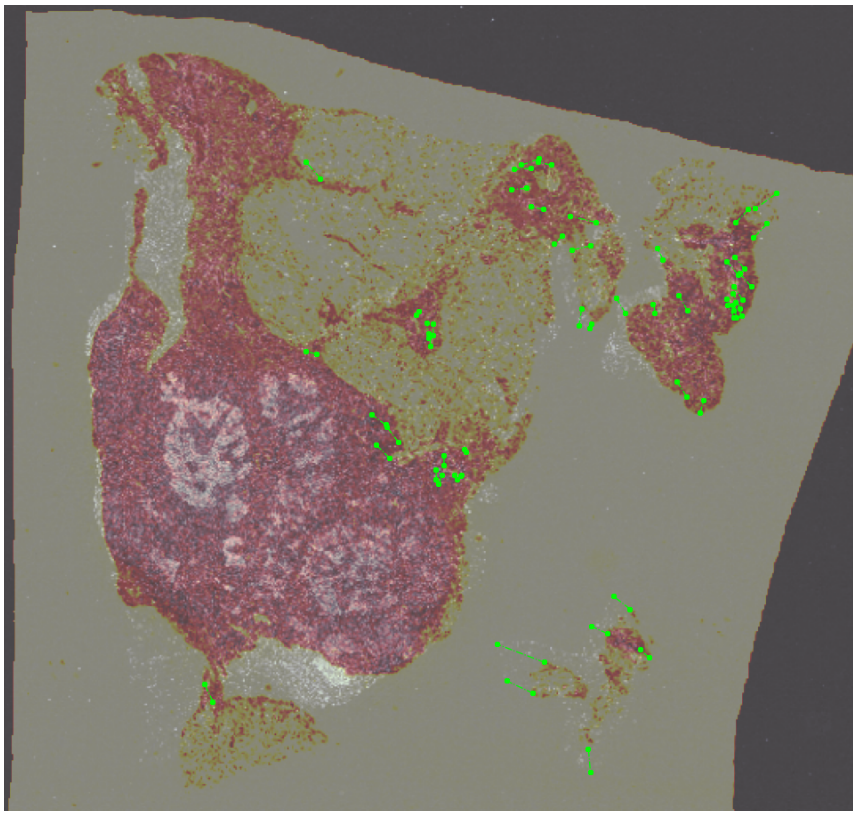
(a) : Original difference



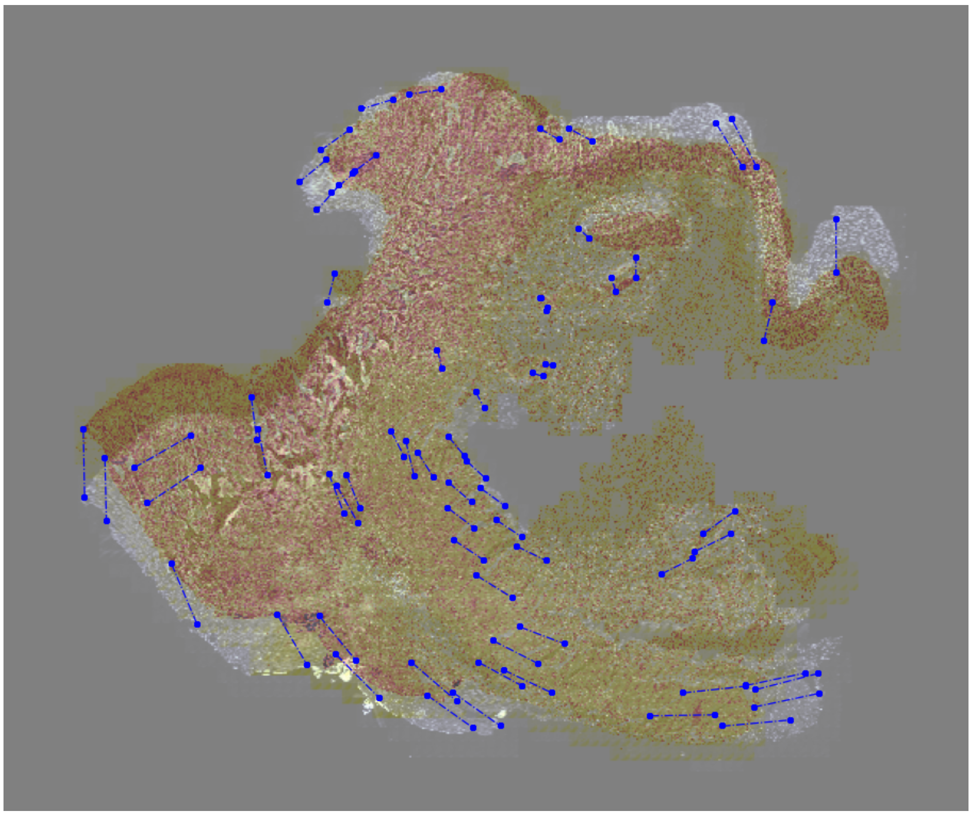
(b) : Warped difference

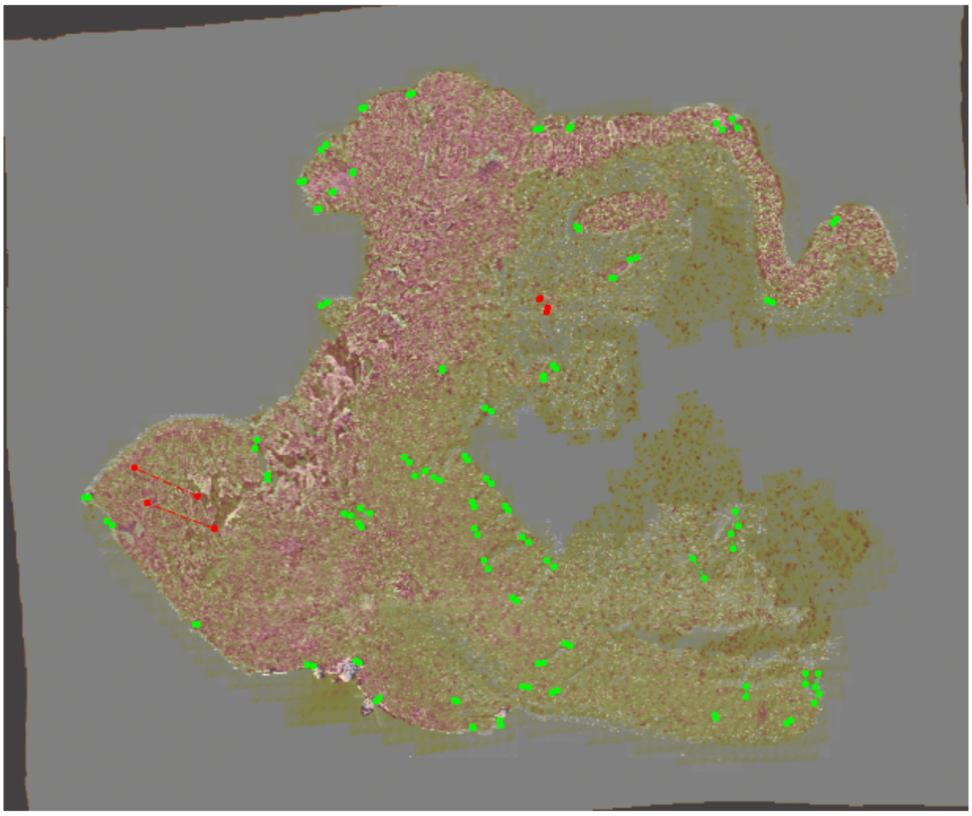


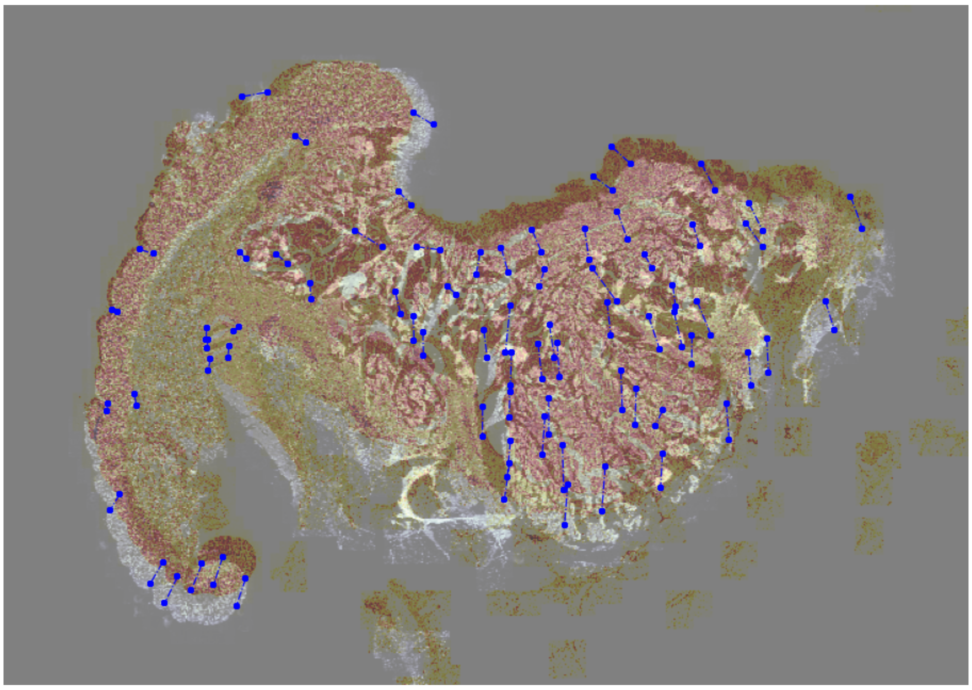
(a) : Original difference

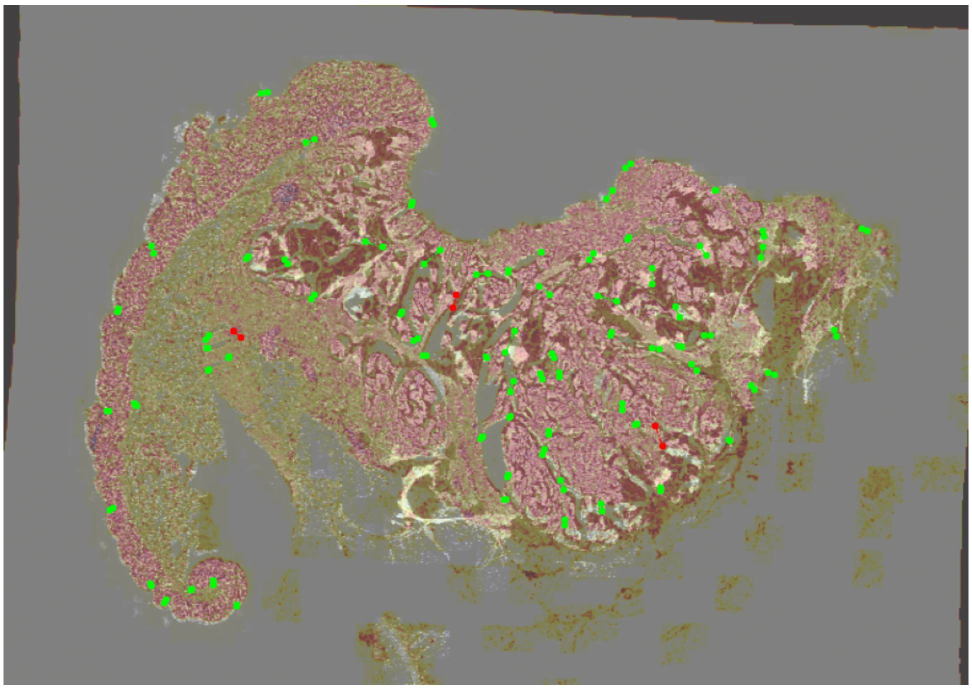


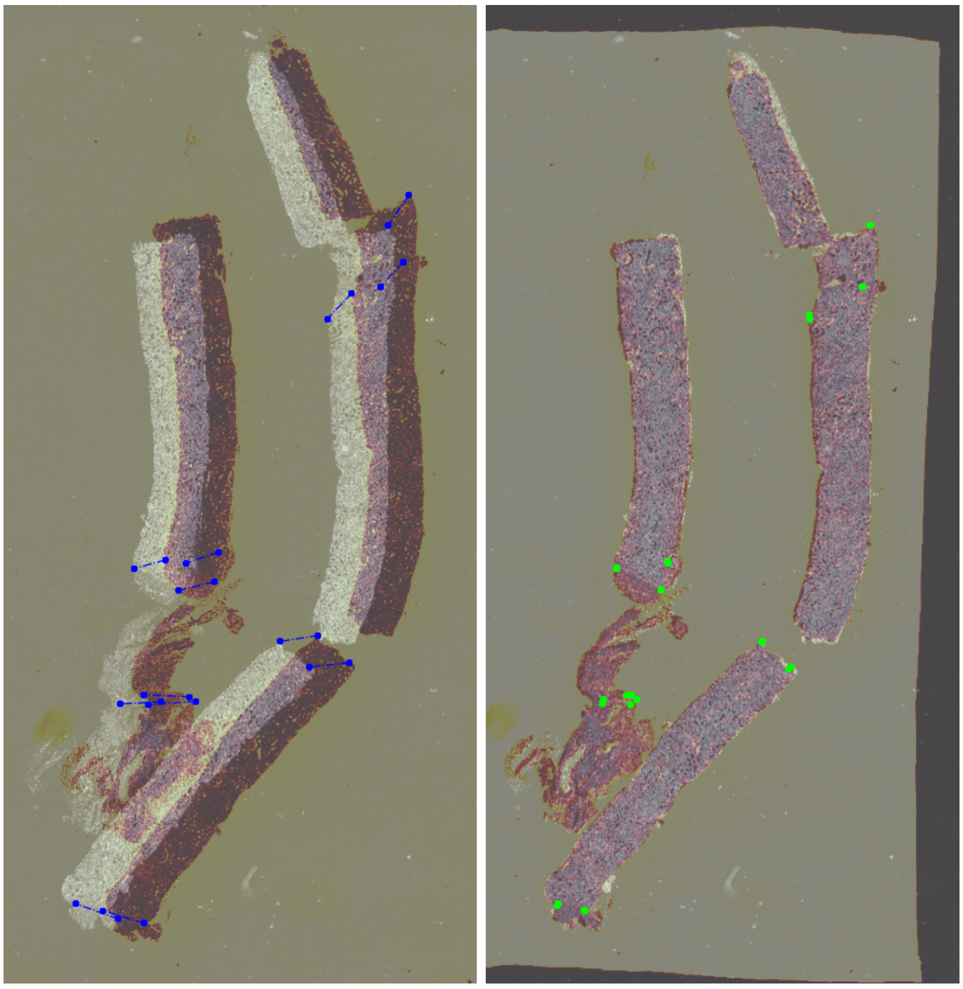
(b) : Warped difference


(a) : Original difference


(b) : Warped difference


(a) : Original difference


(b) : Warped difference



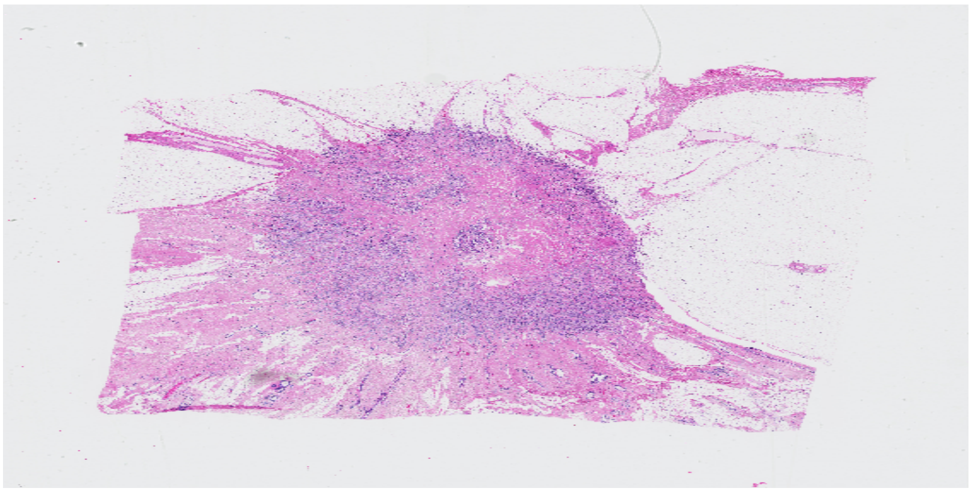
(a) : Original difference **(b) :** Warped difference

9.14 Generalization of our method on another dataset

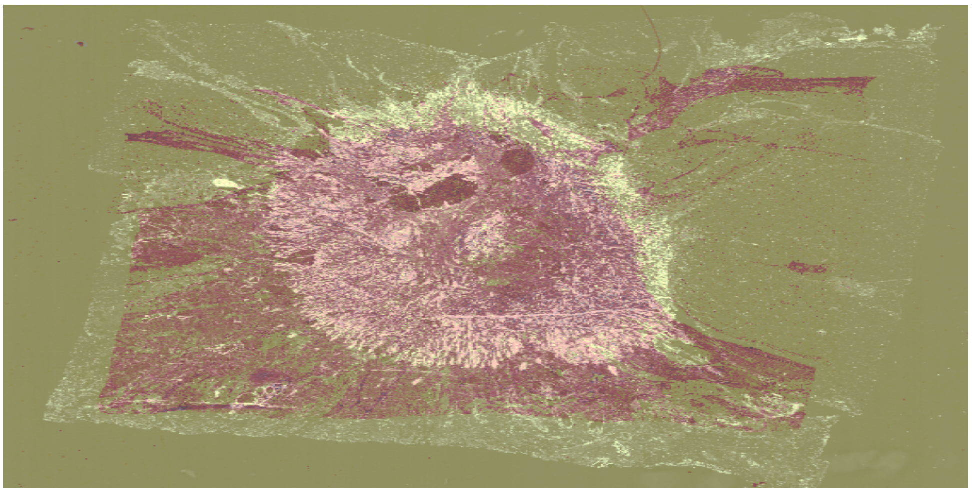
In order to establish whether our method has not overfitted the ANHIR dataset, we have chosen to register several images from the ACROBAT challenge[1]. This dataset contains stained breast cancer tissue samples and therefore is from a similar modality as our ANHIR dataset.

As we can see in the figure 9.24, our method is capable of registering images from different datasets comparably to the ANHIR dataset. The effectiveness of our method was tried only on several images without landmarks. Therefore, further testing is required.

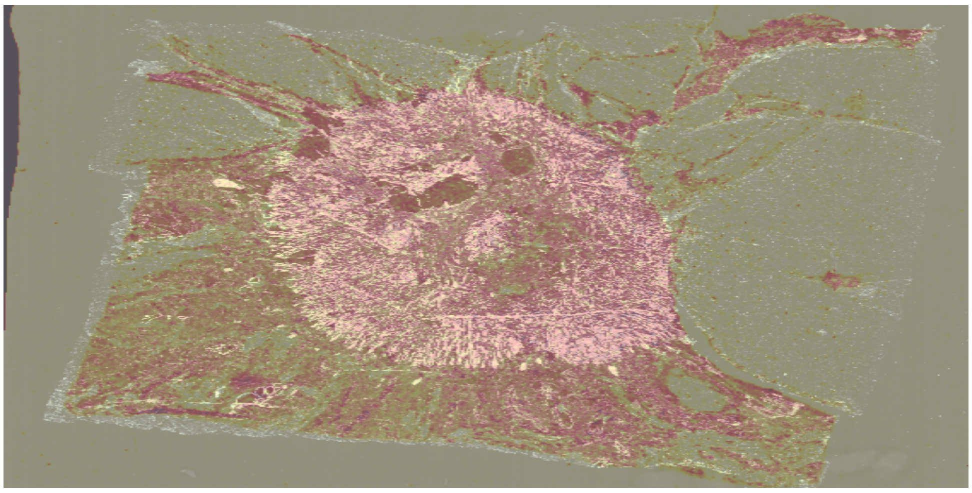
[1]Available from: `https://acrobat.grand-challenge.org/`



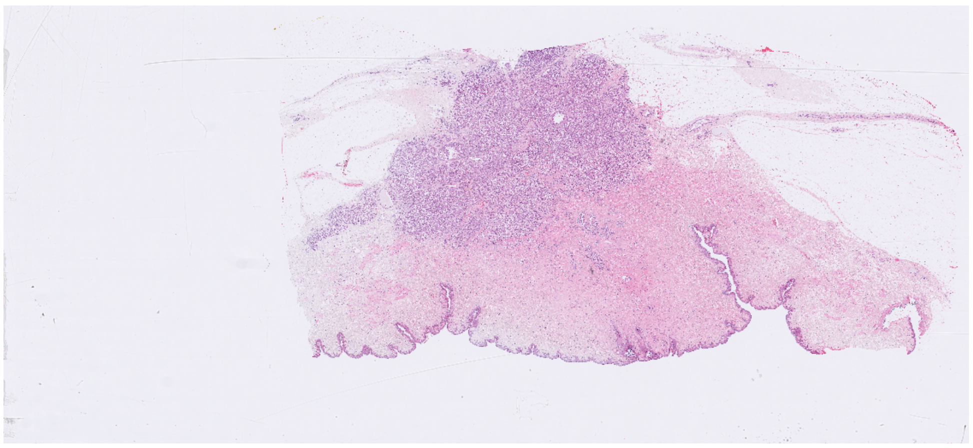
(a) : Original image



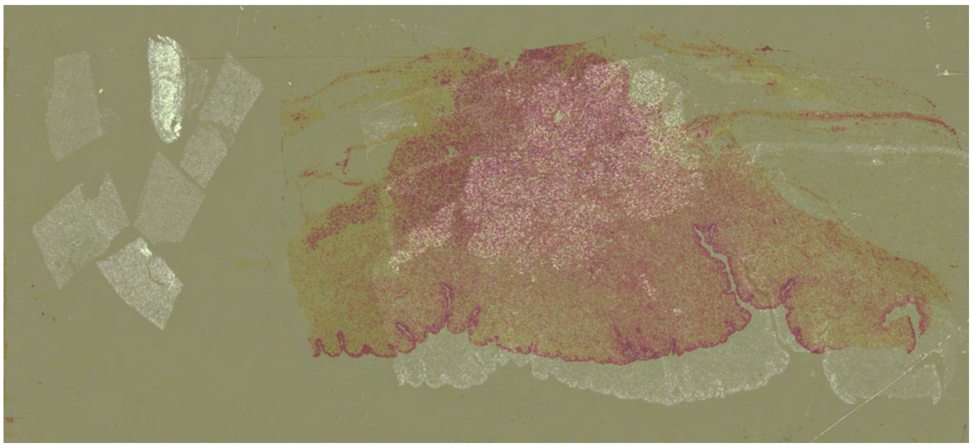
(b) : Original difference



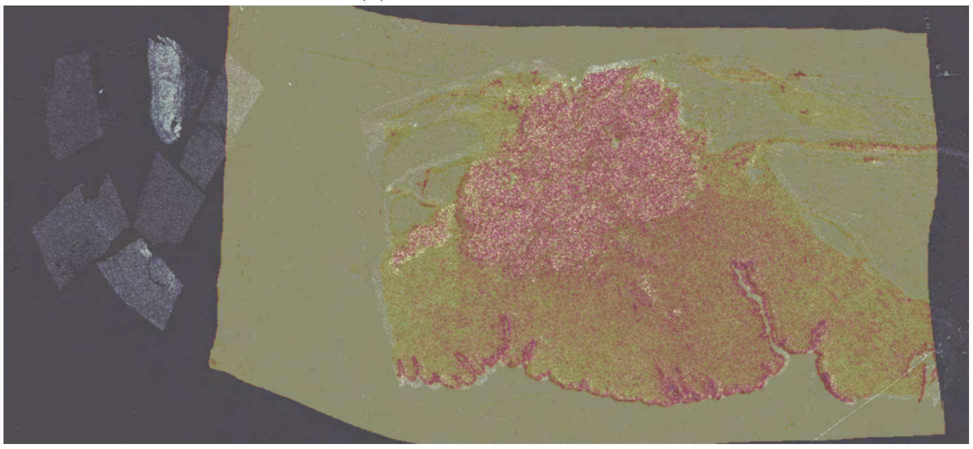
(c) : Warped difference



(d) : Original image



(e) : Original difference



(f) : Warped difference

Figure 9.24: Examples of the registration on the ACROBAT dataset

Chapter 10

Discussion

In this chapter, we will discuss the effectiveness and weaknesses of our registration method, propose possible improvements and report on problems with our training.

10.1 Viability of the optical flow for the registration of histological tissue samples

Optical flow estimation methods have proven to be viable for the histological tissue registration.

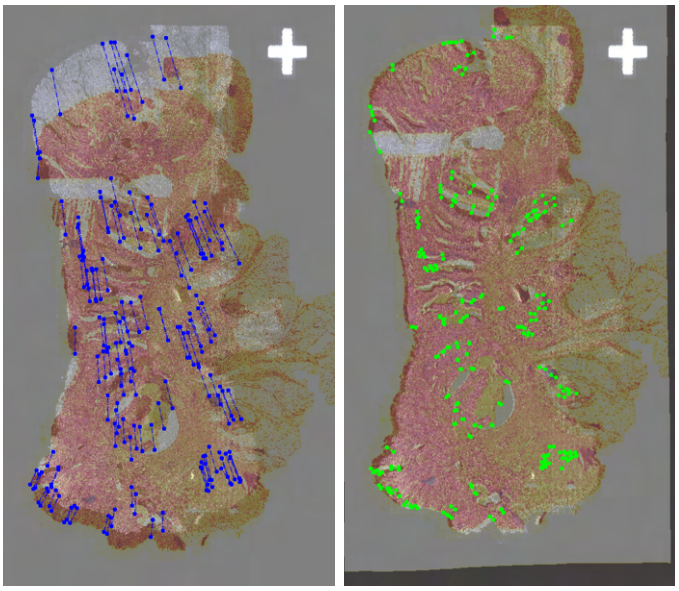
We have shown that our method is capable of both the global pre-alignment and the local registration, given good training data and proper training. The main two assumptions, which are made about the images for the purpose of the optical flow estimation, namely the brightness consistency and local movement homogeneity, can be somewhat met or compensated.

The RAFT-based architectures, which were used in our method, could compensate for the difference in colors between images without encountering any significant issues past the initial few thousand iterations of learning.

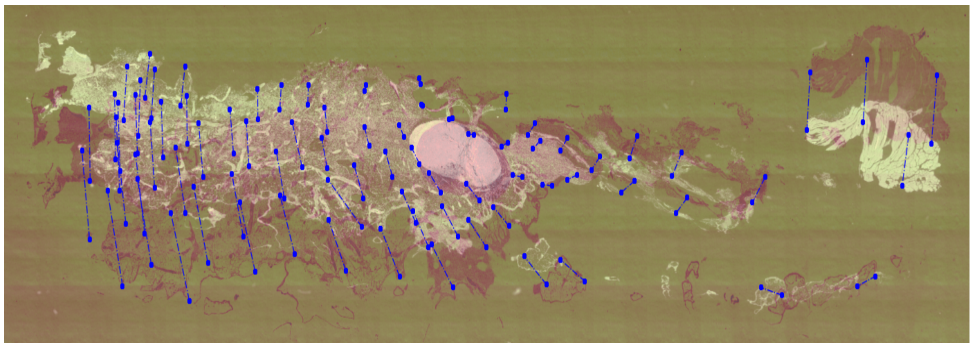
10.2 Encountered problems with the global registration

The ANHIR dataset contains several images with a high magnitude of displacement, and their registration would require a rotation of more than 90 degrees. Additionally, several images, primarily those belonging to the mice kidney sub-dataset, would require scaling before being correctly registered. As seen in section Results, our network is not capable of reaching the same level of accuracy on these images as on the rest of the dataset.

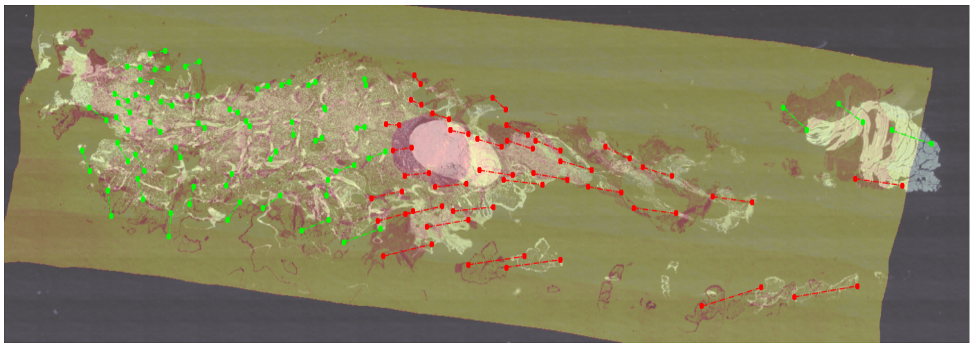
A commonly seen error is the over or under rotation of the image caused by a missing part of the tissue, as shown by the second image pair in figure 10.1. As we have chosen to train the global network only to be capable of registering global motion, its only course of action is to move the entire tissue sample. Since flow generated by this network is smooth, this hugely reduces the change of creation of artifacts in the input image of the local network.



(a) : Original image pair **(b) :** Registered image pair (correct)



(c) : Original image pair

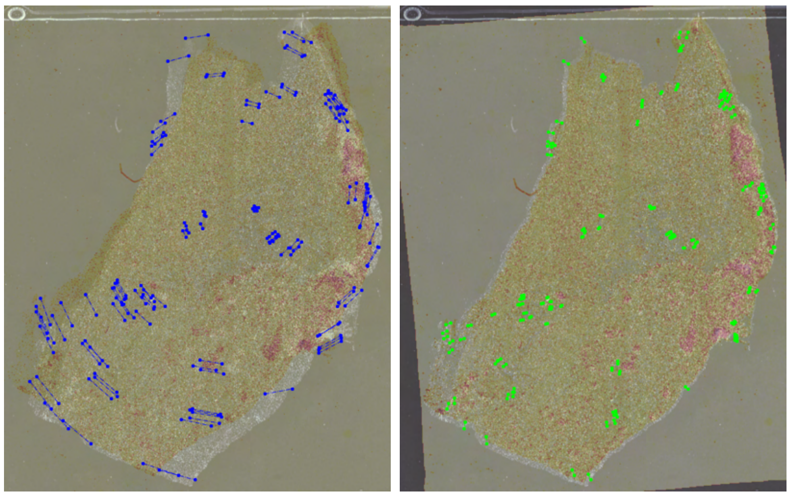


(d) : Registered image pair (incorrect)

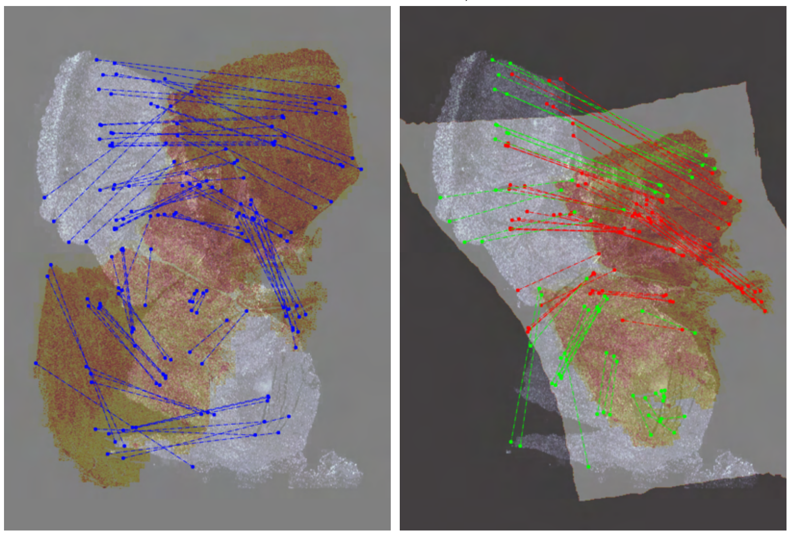
Figure 10.1: Examples of correctly and incorrectly registered images using only the global registration module (blue - initial, green - correct, red - incorrect, correctness in the sense of distance reduction)

It is visible from the second image pair in figure 10.1 that the global network tried to apply a rotation. However, the transformation is more complex, and the registration fails. In the first image pair, the network correctly predicts translation.

10.3 Encountered problems with the local registration



(a) : Original image pair **(b) :** Registered image pair (correct)



(c) : Original image pair **(d) :** Registered image pair (incorrect)

Figure 10.2: Examples of correctly and incorrectly registered images using only the local registration module (blue - initial, green - correct, red - incorrect, correctness in the sense of distance reduction)

The use of the optical flow for local movement faces the same challenges. However, we can only utilize limited downsampling. Furthermore, the iterative flow estimation approach utilized in the RAFT-based architectures diverges after several updates rather than converging or remaining the same.

The network also produces a small amount of error, not generated by a wrong flow estimation but rather by structures in the image itself, which reduces the registration accuracy.

In the figure 10.2 we can see that the network is capable of registering small movements better than the global network. However, it is useless when presented with bigger movements, such as 90-degree rotation.

10.4 Proposed improvements to our method

Our methods essentially utilize a two-level pyramid with varying downsampling and the ability to skip a level when similarity is not increased. However, the original images in the ANHIR dataset have a resolution of tens of thousands of pixels, while we only use images in the size of a few hundred pixels for the global module and around a thousand for the local module, effectively wasting most of the information contained inside the image.

We propose adding additional levels to the pyramid, created in the same way as the first two, with the same similarity check between every two levels and training more networks, which are fine-tuned for the specific resolution. The main problem with adding more higher-resolution layers is the increased memory requirement.

Another possible improvement is connecting the two networks into a single network. Since all the steps in between are capable of a backward pass, creating a single network with a 'bridge' would allow better training, akin to the FlowNet2 architecture. Once again, the main bottleneck is the hardware used since even evaluation would likely require more than double our current memory.

10.5 Issues with the training

The main issue with the training is the small batch size and relatively small image size. For the batch size of 2 and image size of 384×384 for the local network, or a batch size of 4 and image size of 256×256 for the global network, full memory of the GPU (3.8/4 GB) is used. Given access to a more powerful GPU, the increase in both the batch and image size would likely allow the network to achieve better results.

Likely candidates for some of the problems are the several parameters that were chosen arbitrarily, only with limited experimentation. Mainly these are the weights of different unsupervised loss functions, which were chosen only after several short trials, as to see the actual results is heavily time demanding. Giving the right weight to each of the different metrics used for training is essential for successful registration.

Chapter 11

Conclusion

In this thesis, we have developed a method for registering histological tissue samples after they have been stained with the help of optical flow, using convolutional neural network architectures based on the GMA, combining them in a fashion similar to the FlowNet2 architecture. Registration with the help of optical flow has proven to be a viable alternative to the commonly used methods, outperforming the baseline methods and being comparable to the other participants of the ANHIR challenge.

We have created several different training datasets, which were utilized for different parts of the training process and different purposes of the network. The proposed training routine, utilizing both supervised and unsupervised loss functions, has allowed the network to correctly learn the optical flow between the histological image pairs.

The inability of optical flow to register bigger displacements was addressed by creating the global registration module. At the same time, the brightness inconsistency between pixels has been revealed not to be a significant issue, with the network being able to compensate for the difference.

The generalization ability was tried on a second dataset, with images from the same modality as the original ANHIR. We tried only several images, but our method seems to have no issue with their registration.

Bibliography

[1] M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, and B. Yener, "Histopathological image analysis: A review," *IEEE reviews in biomedical engineering*, vol. 2, pp. 147–171, 2009.

[2] K. Suzuki, "Overview of deep learning in medical imaging," *Radiological physics and technology*, vol. 10, no. 3, pp. 257–273, 2017.

[3] J. Pichat, J. E. Iglesias, T. Yousry, S. Ourselin, and M. Modat, "A survey of methods for 3d histology reconstruction," *Medical image analysis*, vol. 46, pp. 73–105, 2018.

[4] M. T. McCann, J. A. Ozolek, C. A. Castro, B. Parvin, and J. Kovacevic, "Automated histology analysis: Opportunities for signal processing," *IEEE Signal Processing Magazine*, vol. 32, no. 1, pp. 78–87, 2014.

[5] C. Ceritoglu, L. Wang, L. D. Selemon, J. G. Csernansky, M. I. Miller, and J. T. Ratnanather, "Large deformation diffeomorphic metric mapping registration of reconstructed 3d histological section images and in vivo mr images," *Frontiers in human neuroscience*, vol. 4, p. 43, 2010.

[6] G. J. Metzger, S. C. Dankbar, J. Henriksen, A. E. Rizzardi, N. K. Rosener, and S. C. Schmechel, "Development of multigene expression signature maps at the protein level from digitized immunohistochemistry slides," *PloS one*, vol. 7, no. 3, p. e33520, 2012.

[7] M. A. Viergever, J. A. Maintz, S. Klein, K. Murphy, M. Staring, and J. P. Pluim, "A survey of medical image registration–under review," 2016.

[8] K. Murphy, B. Van Ginneken, J. M. Reinhardt, S. Kabus, K. Ding, X. Deng, K. Cao, K. Du, G. E. Christensen, V. Garcia, *et al.*, "Evaluation of registration methods on thoracic ct: the empire10 challenge," *IEEE transactions on medical imaging*, vol. 30, no. 11, pp. 1901–1920, 2011.

[9] J. Borovec, J. Kybic, I. Arganda-Carreras, D. V. Sorokin, G. Bueno, A. V. Khvostikov, S. Bakas, E. I. Chang, S. Heldmann, K. Kartasalo, L. Latonen, J. Lotz, M. Noga, S. Pati, K. Punithakumar, P. Ruusuvuori, A. Skalski, N. Tahmasebi, M. Valkonen, L. Venet, Y. Wang, N. Weiss,

M. Wodzinski, Y. Xiang, Y. Xu, Y. Yan, P. Yushkevich, S. Zhao, and A. Munoz-Barrutia, “Anhir: Automatic non-rigid histological image registration challenge,” *IEEE transactions on medical imaging*, vol. 39, no. 10, pp. 3042–3052, 2020.

[10] J. Borovec, A. Munoz-Barrutia, and J. Kybic, “Benchmarking of image registration methods for differently stained histological slides,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 3368–3372, IEEE, 2018.

[11] D. Fleet and Y. Weiss, “Optical flow estimation,” in *Handbook of mathematical models in computer vision*, pp. 237–257, Springer, 2006.

[12] J. G. Black and L. J. Black, *Microbiology: principles and explorations*. John Wiley & Sons, 2018.

[13] H. A. Alturkistani, F. M. Tashkandi, and Z. M. Mohammedsaleh, “Histological stains: a literature review and case study,” *Global journal of health science*, vol. 8, no. 3, p. 72, 2016.

[14] M. Titford, “Progress in the development of microscopical techniques for diagnostic pathology,” *Journal of Histotechnology*, vol. 32, no. 1, pp. 9–19, 2009.

[15] S. Shostak, “Histology’s nomenclature: Past, present and future,” *Biol Syst Open Access*, vol. 2, no. 122, p. 2, 2013.

[16] G. Musumeci, “Past, present and future: overview on histology and histopathology,” *J Histol Histopathol*, vol. 1, no. 5, pp. 1–3, 2014.

[17] M. H. Ross and W. Pawlina, *Histology*. Lippincott Williams & Wilkins, 2006.

[18] K. Loverdos, A. Fotiadis, C. Kontogianni, M. Iliopoulou, and M. Gaga, “Lung nodules: A comprehensive review on current approach and management,” *Annals of Thoracic Medicine*, vol. 14, no. 4, p. 226, 2019.

[19] C. P. Hansen, H. Holtveg, D. Francis, L. Rasch, and S. Bertelsen, “Pulmonary hamartoma,” *The Journal of Thoracic and Cardiovascular Surgery*, vol. 104, no. 3, pp. 674–678, 1992.

[20] K. Zaman, “Tuberculosis: a global health problem,” *Journal of health, population, and nutrition*, vol. 28, no. 2, p. 111, 2010.

[21] D. Ost, A. M. Fein, and S. H. Feinsilver, “The solitary pulmonary nodule,” *New England Journal of Medicine*, vol. 348, no. 25, pp. 2535–2542, 2003.

[22] A. B. Shinagare, G. Cunto-Amesty, and F. M. Fennessy, “Multiple inflammatory nodules: a differential diagnosis of new pulmonary nodules in oncology patients,” *Cancer Imaging*, vol. 10, no. 1, p. 205, 2010.

[23] L. T. Tanoue, N. T. Tanner, M. K. Gould, and G. A. Silvestri, “Lung cancer screening,” *American journal of respiratory and critical care medicine*, vol. 191, no. 1, pp. 19–33, 2015.

[24] C. G. Irvin and J. H. Bates, “Measuring the lung function in the mouse: the challenge of size,” *Respiratory research*, vol. 4, no. 1, pp. 1–9, 2003.

[25] G. Betts, K. Young, J. Wise, E. Johnson, B. Poe, D. Kruse, O. Korol, J. Johnson, M. Womble, and P. DeSaix, “Anatomy & physiology - gross anatomy of the lungs,” 2013. [Online; accessed April 6, 2022].

[26] W. Commons, “Normal appearance of elastic laminae of visceral pleura. elastic tissue stain.,” 2009. [Online; accessed April 7, 2022].

[27] B. Weigelt, F. C. Geyer, and J. S. Reis-Filho, “Histological types of breast cancer: How special are they?,” *Molecular Oncology*, vol. 4, no. 3, pp. 192–208, 2010. Thematic Issue: The Molecular Biology of Breast Cancer.

[28] R. G. do Nascimento and K. M. Otoni, “Histological and molecular classification of breast cancer: what do we know,” *Mastology*, vol. 30, pp. 1–8, 2020.

[29] A. E. M. Reed, J. R. Kutasovic, S. R. Lakhani, and P. T. Simpson, “Invasive lobular carcinoma of the breast: morphology, biomarkers and’omics,” *Breast cancer research*, vol. 17, no. 1, pp. 1–11, 2015.

[30] L. M. Biga, S. Dawson, A. Harwell, R. Hopkins, J. Kaufmann, M. LeMaster, P. Matern, K. Morrison-Graham, D. Quick, and J. Runyeon, *Anatomy & physiology*. OpenStax & Oregon State University, 2020.

[31] H. H. Hartgrink, E. P. Jansen, N. C. van Grieken, and C. J. van de Velde, “Gastric cancer,” *The Lancet*, vol. 374, no. 9688, pp. 477–490, 2009.

[32] E. C. Smyth, M. Nilsson, H. I. Grabsch, N. C. van Grieken, and F. Lordick, “Gastric cancer,” *The Lancet*, vol. 396, no. 10251, pp. 635–648, 2020.

[33] A. S. Levey and J. Coresh, “Chronic kidney disease,” *The lancet*, vol. 379, no. 9811, pp. 165–180, 2012.

[34] G. Singh and S. L. Katyal, “Clara cells and clara cell 10 kd protein (cc10),” *American journal of respiratory cell and molecular biology*, vol. 17, no. 2, pp. 141–143, 1997.

[35] J. K. Chan, “The wonderful colors of the hematoxylin–eosin stain in diagnostic surgical pathology,” *International journal of surgical pathology*, vol. 22, no. 1, pp. 12–32, 2014.

[36] T. Scholzen and J. Gerdes, “The ki-67 protein: from the known and the unknown,” *Journal of cellular physiology*, vol. 182, no. 3, pp. 311–322, 2000.

[37] H. M. DeLisser, P. J. Newman, and S. M. Albelda, “Molecular and functional aspects of pecam-1/cd31,” *Immunology Today*, vol. 15, no. 10, pp. 490–495, 1994.

[38] M. Hung and Y. Lau, “Basic science of her-2/neu: a review,” *Seminars in oncology*, vol. 26, p. 51—59, August 1999.

[39] C. M. Klinge, “Estrogen receptor interaction with estrogen response elements,” *Nucleic acids research*, vol. 29, no. 14, pp. 2905–2919, 2001.

[40] E. Vegeto, M. M. Shahbaz, D. X. Wen, M. E. Goldman, B. W. O’Malley, and D. P. McDonnell, “Human progesterone receptor a form is a cell-and promoter-specific repressor of human progesterone receptor b function.,” *Molecular endocrinology*, vol. 7, no. 10, pp. 1244–1255, 1993.

[41] S. L. Grimm, S. M. Hartig, and D. P. Edwards, “Progesterone receptor signaling mechanisms,” *Journal of Molecular Biology*, vol. 428, no. 19, pp. 3831–3849, 2016. Molecular Basis of Signal Transduction.

[42] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” 2015.

[43] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, “A naturalistic open source movie for optical flow evaluation,” in *European Conf. on Computer Vision (ECCV)* (A. Fitzgibbon et al. (Eds.), ed.), Part IV, LNCS 7577, pp. 611–625, Springer-Verlag, Oct. 2012.

[44] B. Zitová and J. Flusser, “Image registration methods: a survey,” *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[45] X. Chen, A. Diaz-Pinto, N. Ravikumar, and A. F. Frangi, “Deep learning in medical image registration,” *Progress in biomedical engineering (Bristol)*, vol. 3, no. 1, 2021.

[46] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–1402, Ieee, 2003.

[47] L. M. Fonseca and B. Manjunath, “Registration techniques for multisensor remotely sensed imagery,” *PE & RS- Photogrammetric Engineering & Remote Sensing*, vol. 62, no. 9, pp. 1049–1056, 1996.

[48] P. Viola and W. M. Wells III, “Alignment by maximization of mutual information,” *International journal of computer vision*, vol. 24, no. 2, pp. 137–154, 1997.

[49] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, “Multimodality image registration by maximization of mutual information,” *IEEE transactions on Medical Imaging*, vol. 16, no. 2, pp. 187–198, 1997.

[50] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, “Pyramid methods in image processing,” *RCA engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[51] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, “Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume,” 2018.

[52] M. Fornefett, K. Rohr, and H. S. Stiehl, “Radial basis functions with compact support for elastic registration of medical images,” *Image and vision computing*, vol. 19, no. 1-2, pp. 87–96, 2001.

[53] F. L. Bookstein, “Principal warps: Thin-plate splines and the decomposition of deformations,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[54] B. K. Horn and B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981.

[55] S. S. Beauchemin and J. L. Barron, “The computation of optical flow,” *ACM Comput. Surv.*, vol. 27, p. 433–466, sep 1995.

[56] M. J. Black and A. D. Jepson, “Eigentracking: Robust matching and tracking of articulated objects using a view-based representation,” *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[57] Z. Teed and J. Deng, “Raft: Recurrent all-pairs field transforms for optical flow,” in *European conference on computer vision*, pp. 402–419, Springer, 2020.

[58] P. Nesi, “Variational approach to optical flow estimation managing discontinuities,” *Image and Vision Computing*, vol. 11, no. 7, pp. 419–439, 1993.

[59] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, “Performance of optical flow techniques,” *International journal of computer vision*, vol. 12, no. 1, pp. 43–77, 1994.

[60] A. Ranjan and M. J. Black, “Optical flow estimation using a spatial pyramid network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4161–4170, 2017.

[61] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470, 2017.

[62] S. Zhao, Y. Sheng, Y. Dong, E. I.-C. Chang, and Y. Xu, "Maskflownet: Asymmetric feature matching with learnable occlusion mask," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[63] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," 2017.

[64] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9772–9781, 2021.

[65] D. Coltuc, P. Bolon, and J.-M. Chassery, "Exact histogram specification," *IEEE Transactions on Image processing*, vol. 15, no. 5, pp. 1143–1152, 2006.

[66] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006, p. 1100612, International Society for Optics and Photonics, 2019.

[67] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[68] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[69] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2018.

[70] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015.

[71] H. Gholamalinezhad and H. Khosravi, "Pooling methods in deep neural networks, a review," 2020.

[72] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.

[73] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[74] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[75] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," *CoRR*, vol. abs/1910.05446, 2019.

[76] S. ichi Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4, pp. 185–196, 1993.

[77] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[78] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[79] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, vol. 9351, pp. 234–241. Cham: Springer International Publishing, 2015.

[80] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[81] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in neural information processing systems*, vol. 27, 2014.

Appendix A

List of abbreviations and acronyms

Adam	**Adap**tive **M**oment Estimation
AGT	**A**lternate-**g**roup **t**ransformer
AIDPATH	**A**rtificial **I**ntelligence-driven, **D**ecentralized **P**roduction for **A**dvanced **T**herapies in the **H**ospital
AMrTRE	**A**verage **M**edian **rTRE**
AMxrTRE	**A**verage **M**aximum **rTRE**
ANHIR	**A**utomatic **N**on-rigid **H**istological **I**mage **R**egistration
ARMrTRE	**A**verage **R**ank of **M**edian **rTRE**
ARMxrTRE	**A**verage **R**ank **M**aximum **rTRE**
AsymOFMM	**Asym**metric **O**cclusion-Aware **F**eature **M**atching **M**odule
AWGN	**A**dditive **W**hite **G**aussian **N**oise
BN	**B**atch **N**orm
BRCA	**Br**east **Ca**ncer
CAD	**C**omputer **A**ssisted **D**iagnosis
CC10	**C**lara **C**ell **10**
CD	**C**luster of **D**ifferation
CDF	**C**umulative **D**ensity **F**unction
CE	**C**ross **E**ntropy
CIMA	**C**enter for **A**pplied **M**edical **R**esearch
CKD	**C**hronical **K**idney **D**isease
CN	**C**ontext **N**etwork
CNN	**C**onvolutional **N**eural **N**etworks
COAD	**Co**lon **Ad**enocarcinoma
CP	**C**ontrol **P**oint
CT	**C**omputed **T**omography
DNA	**D**eoxyribo**n**ucleic **A**cid
EM	**E**stimation **M**aximalization
EPE	**E**nd **P**oint **E**rror
ER	**E**strogen **R**ecopter
FN	**F**eature **N**etwork
FP	**F**eature **P**yramid
FPN	**F**eature **P**yramid **N**etwork
GD	**G**radient **D**escent
GT	**G**round **T**ruth

GRU	**G**ated **R**ecurrent **U**nit
H&E	**H**ematoxylin and **E**osin
HER2	**H**uman **E**pidermal Growth Factor **R**eceptor **2**
HSV	**H**ue, **S**aturation, **V**alue
IDC	**I**nvasive **D**uctal **C**arcinoma
IDC-NST	**I**nvasive **D**uctal **C**arcinoma **N**o **S**pecific **T**ype
ILC	**I**nvasive **L**obular **C**arcinoma
iUS	**I**ntra-operative **U**ltra**s**ound
kNN	**k** - **N**earest **N**eighbours
LR	**L**earning **R**ate
LS	**L**east **S**quares
LReLU	**L**eaky **Re**ctified **L**inear **U**nit
MAE	**M**ean **A**bsolute **E**rror
MD	**M**aximum **D**isplacement
MI	**M**utual **I**nformation
ML	**M**achine **L**earning
MMrTRE	**M**edian of **M**edian **rTRE**
MRI	**M**agnetic **R**esonance **I**maging
mRNA	**M**essenger **R**ibo**n**ucleic **A**cid
MSE	**M**ean **S**quared **E**rror
NBF	**N**eutral **B**uffered **F**ormalin
NN	**N**eural **N**etwork
PAS	**P**eriodic **A**cid-**S**chiff
PDF	**P**robability **D**ensity **F**unction
PECAM-1	**P**latelet **en**dothelial **c**ell **a**dhesion **m**olecule **1**
PET	**P**ositron **E**mission **T**omography
PR	**P**rogesterone **r**eceptor
PReLU	**P**aremetric **Re**ctified **L**inear **U**nit
proSPC	**Pro**surfactant **P**rotein **C**
RBF	**R**adial **B**asis **F**unction
RAFT	**R**ecurrent **A**ll-Pairs **F**ield **T**ransforms
ReLU	**Re**ctified **L**inear **U**nit
ResNet	**Res**idual **Net**work
RIRE	**R**etrospective **I**ntermodality Brain Image **Re**gistraion
rTRE	**r**elative **T**arget **R**egistration **E**rror
SGD	**S**tochastic **G**radient **D**escent
SSIM	**S**tructural **S**imilarity **I**ndex **M**easure
SSD	**S**olid **S**tate **D**isk
SMA	**S**mooth **M**uscle **A**ctin
SpyNet	**S**patial **Py**ramid **Net**work
TPS	**T**hin-**P**late **s**plines
VGG	**V**isual **G**eometry **G**roup
UCLM	**U**niversidad de **C**astilla-**L**a **M**ancha

Appendix B

Neural networks

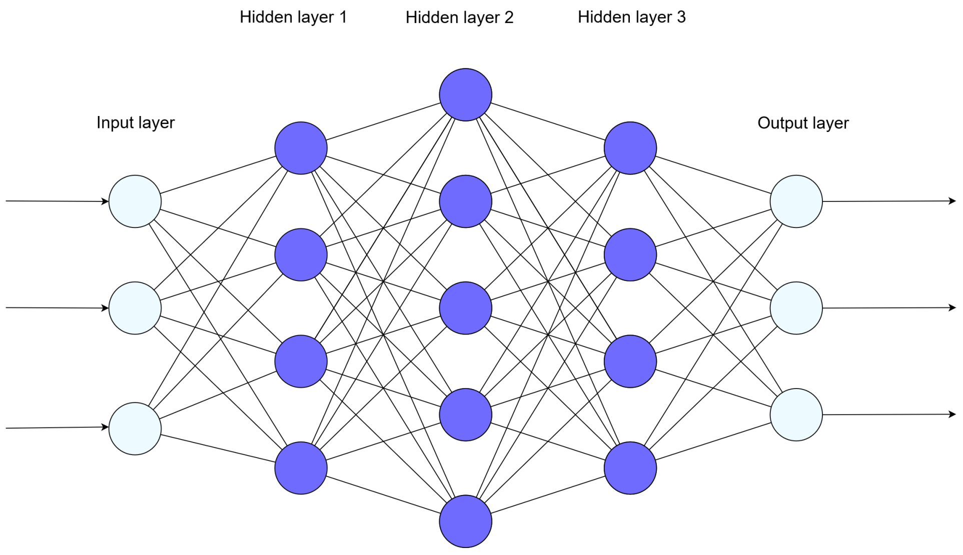


Figure B.1: Basic structure of a feedforward neural network

Neural networks are helpful for a wide variety of different tasks, from time series classification, recommendation creation, text analyzing, and most importantly, in our case, image processing [68]. The purpose of the network is to find an approximation of a function f^*, that defines a mapping from input space to output space such that $y = f^*(\mathbf{x}; \theta)$, where $\mathbf{x}$ is the input of the network, usually in a shape of a data grid, y is the output with variable shape. θ are values of the parameters of the hidden layers of the network (represented by the individual connections between the nodes in B.1) that result in the best approximation of the unknown function [68].

This section discusses essential components of a neural network, such as a convolution layer, a fully-connected layer, and their basic structure.

B.1 Convolution layer

The vital building block of any CNN is a convolutional layer. These layers focus on learning and using the convolution kernels. Values of each kernel K are learned from the specific features of the input I_I [69]. These kernels are way smaller than the respective image they are used on, usually around several pixels in height and width. The space from which they can receive information is called a receptive field [70]. As they move through the input matrix, output I_O is computed at each position as

$$I_O(i,j) = \sigma\left(b + \sum_{(m,n)\in K} I_I(i+m, j+n)K(m,n)\right). \tag{B.1}$$

where b is the bias and σ is some non-linearity [70].

We see that output values correspond only to several input values and, as such, vastly reduce the number of parameters of the network compared to a fully-connected layer. Convolution reduces the overall size of the input at the edges. Padding prevents the image's size from changing by adding some value to the edge of the image (usually zeros). We can also reduce the output size further by increasing the stride of the kernel, which makes it, so we compute convolution on only some of the input values [70] such that

$$I_O(m+1, n+1) = \mathrm{Conv}(I_I(x+s_x, y+s_y), K), \tag{B.2}$$

where $s_{(\cdot)}$ is the stride in given direction. This also allows the network to find connections between areas that are not directly bordering one another, which is ideal for image registration, as we can use this fact to construct a feature pyramid.

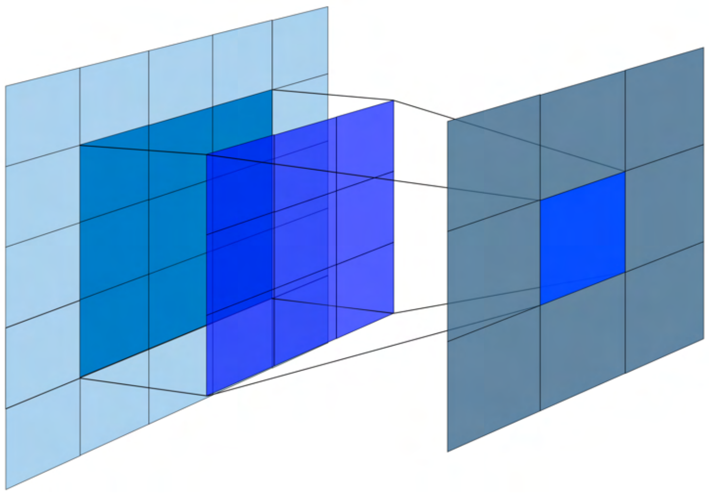


Figure B.2: Convolution with 3×3 kernel

B.2 Fully-connected layer

As the name implies, a fully-connected layer, sometimes called a dense layer (or linear layer), consists of neurons connected to every output of the previous layer via a connection with learned weight. These layers are usually used to map outputs of convolutional layers into the desired output. They are shown in figure B.1 as the hidden layers. Since the fully-connected layer does not differentiate between the relative positions of each of the inputs, the equation can be rewritten in 1D. The output of such layer is computed as

$$I_O(i) = \sigma\left(b_i + \sum_{j=0}^{n} w_{i,j} I_I(j)\right), \tag{B.3}$$

where b is the bias, σ some non-linearity, w are the weights of each connection from the inputs I_I to the outputs I_O [68].

B.3 Pooling layer

Pooling layers allow neural networks to aggregate information over space. They are usually used after a convolution layer to reduce the dimensionality of the feature maps. Using pooling can also provide a form of spatial transformation invariance or remove unnecessary details. The pooling filter of a given size and stride moves throughout the input image [71].
Two pooling layers, which are commonly used, are a max-pooling layer and an average-pooling layer. Max pooling outputs only the highest value from the given receptive field. This can be written for each output value as

$$I_O(i,j) = \max_{(m,n)\in A(i,j)} I_I(m,n), \tag{B.4}$$

where $A(i,j)$ is the receptive field of the output [71].
Average pooling returns the average of values from the receptive field, which can be written for each of the output values as

$$I_O(i,j) = \frac{1}{|A(i,j)|} \sum_{(m,n)\in A(i,j)} I_I(m,n), \tag{B.5}$$

where $|A_{i,j}|$ is the number of elements in the receptive field [71].

B.4 Activation functions

Activation functions are used to determine whether the neuron should fire or not. They are usually non-linear as we need non-linear mapping to estimate some functions (e.g., XOR function). A considerable variety of activation functions exist, each with a different place in the overall NN architecture [72].

B.4.1 ReLU

Rectified linear unit, or ReLU for short, is the most commonly used activation function today. The ReLU implements a threshold operation against zero and is formulated as follows

$$f(x) = \max(0, x). \tag{B.6}$$

It provides faster learning and better performance than other previously used functions such as sigmoid or tanh. However, ReLU easily overfits, and due to the zero output value in the negative interval, the gradient from such neurons causes them to become inactive. There is also an issue of gradient discontinuity for the zero input value. The most commonplace location of the ReLU function is after a convolution or a fully-connected layer. ReLU gives mapping to interval $[0, \infty)$ [72].

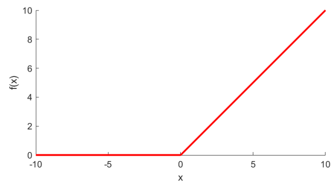


Figure B.3: ReLU

Leaky ReLU

Leaky ReLU addresses the problem with the inactive neurons. Instead of output becoming zero, it is multiplied by some small constant. This constant can also be learnable. We call such function Parametric ReLU (PReLU). The equation is

$$f(x) = \max(0, x) + a \min(0, x), \tag{B.7}$$

where a is some small constant [72].

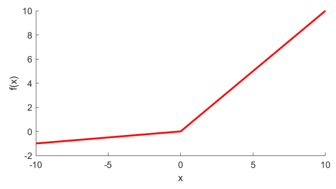


Figure B.4: Leaky ReLU

Softplus

Softplus is a smooth function without zero gradient problems. Therefore it is more stable than standard ReLU [72]. It is defined as

$$f(x) = \log(1 + \exp x). \tag{B.8}$$

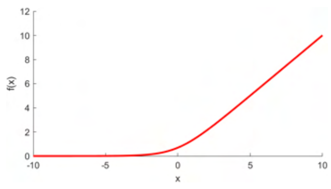


Figure B.5: Softplus

B.4.2 Sigmoid

Sigmoid is usually used at the end of neural networks. Compared to ReLU, it is differentiable for all real-valued inputs. Issues arise when the value moves further away from 0.5, as the gradient becomes progressively smaller. Its mapping to $[0, 1]$ interval is ideal for binary classification [72]. Sigmoid is defined as

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{B.9}$$

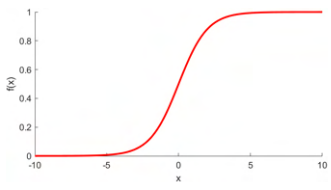


Figure B.6: Sigmoid

B.4.3 Tanh

The hyperbolic tangent function is a zero-centered function that maps to the $[-1, 1]$ interval and, as such, is ideal for the output layer of the neural network. Compared to the sigmoid function, its learning process is faster [72]. Tanh is given by

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}. \tag{B.10}$$

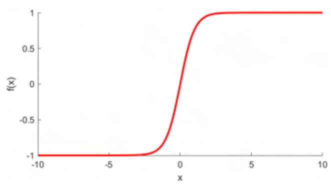


Figure B.7: Hyperbolic tangent

B.4.4 Softmax

The softmax activation function is used in multi-class models to return the probability of each class as

$$f(x_i) = \frac{\exp(x_i)}{\sum_{j=0}^{N} \exp(x_j)}, \tag{B.11}$$

with the resulting class being $q^* = \underset{i}{\operatorname{argmin}} f(x_i)$ [72].

B.5 Loss functions

The loss function provides us with necessary values for learning, as it helps establish how close we are to the desired output. There are many different types of loss functions, each more suitable for the given tasks in the neural network architecture than the other [68]. Some of the most commonly used loss functions are

mean absolute error (MAE),

$$L_{MAE} = \frac{1}{N} \sum_{i=0}^{N} |y_i - \hat{y}_i|, \tag{B.12}$$

mean squared error (MSE),

$$L_{MSE} = \frac{1}{N} \sum_{i=0}^{N} ||y_i - \hat{y}_i||^2, \tag{B.13}$$

and cross-entropy loss (CE),

$$L_{CE} = -\sum_{i=1}^{N} p_i log(p_i). \tag{B.14}$$

Other more specialized types exist such as sequential loss, which uses the list of optical flow estimates produced by some optical flow DNN architectures. Flows are weighted, so later updates are more important than the first ones. Loss is defined as

$$L_{SEQ} = \sum_{i=1}^{N} \gamma^{N-i} \|F_{gt} - F_i\|_1 \tag{B.15}$$

where F_{gt} is the flow ground truth, F_i are the network outputs and $\gamma \in [0, 1]$ [57].

B.6 Other layer types

B.6.1 Batch Normalization

Batch normalization (BN) is used for regularization and stabilization while also reducing the network's dependence on the initialization of parameters by making it so the values of the layer within a given batch have zero mean and unit variance. This is done by performing normalization [73] for each mini-batch B such that

$$\hat{B} = \frac{B - \overline{B}}{\sigma_B}. \tag{B.16}$$

where σ_B is the standard deviation.

B.6.2 Dropout

Dropout prevents the network from overfitting. During training, we randomly set a large subset of parameters to zeros (usually 50%). The dropout layers are turned off during testing. One of the drawbacks is the increase in the learning time [74].

B.7 Optimizer

Selecting an optimizer is one of the most critical steps in optimization of a neural network. The purpose of the optimizer is to minimize the loss function l by updating the parameters of the network. The update is generally done by finding a cost function gradient and taking a step in this direction [75].

B.7.1 SGD

Stochastic gradient descent (SGD) is one of the simplest methods for learning the ideal parameters of the NN. It's update rule is

$$\theta_{t+1} = \theta_t - \alpha_t \nabla L(\theta_t) \tag{B.17}$$

where θ is set of parameters and α is the learning rate [76]. We can improve this simple method by introducing momentum, which takes into account gradient of previous iterations. Update rule is then as follows

$$\begin{aligned} v_0 &= 0 \\ v_{t+1} &= \gamma v_t + \nabla L(\theta_t) \\ \theta_{t+1} &= \theta_t - \alpha_t v_{t+1} \end{aligned} \tag{B.18}$$

where γ is the momentum rate.
Another space for improvement is a stochastic approach (SGD). Given large enough batch of randomly chosen entries from the dataset, it is possible to approximately calculate the loss function from a single batch and thus saving memory in the process (on larger datasets, standard GD may not be even possible). Another version of SGD with momentum is so called SGD Nesterov [76], which is defined as

$$\begin{aligned} v_0 &= 0 \\ v_{t+1} &= \gamma v_t + \nabla L(\theta_t) \\ \theta_{t+1} &= \theta_t - \alpha_t(\gamma v_{t+1} + \nabla L(\theta_t)) \end{aligned} \tag{B.19}$$

B.7.2 Adam

Adam's name is derived from adaptive moment estimation. This method computed individual learning rates of different parameters from first and second moments of the gradient. Some of its advantages are good performance on sparse gradients or that the magnitudes of parameter updates are invariant to gradient rescaling [77]. Adam algorithm is defined as

$$\begin{aligned}
m_0 &= 0, \quad v_0 = 0 \\
m_{t+1} &= \beta_1 m_t + (1-\beta_1)\nabla L(\theta_t) \\
v_{t+1} &= \beta_2 v_t + (1-\beta_2)\nabla L(\theta_t)^2 \\
b_{t+1} &= \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}} \\
\theta_{t+1} &= \theta_t - \alpha_t \frac{m_{t+1}}{\sqrt{v_{t+1}}+\epsilon} b_{t+1}
\end{aligned} \tag{B.20}$$

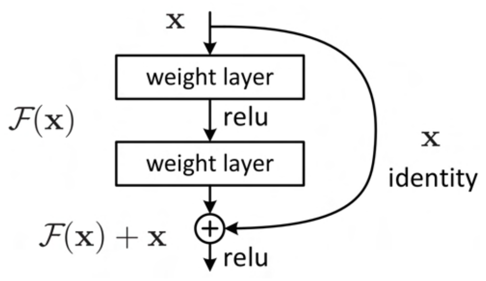
where α is the learning rate, β are exponential decay rates, m is first moment, v is second moment [75].

B.8 Encoder - decoder CNN

These types of networks consist of two parts. Encoder, where the network learns to find features from the input image, and decoder, which maps said features into an output such as classification, segmentation, or in our case, registration [45].

B.8.1 ResNet

Residual networks, or ResNets [78] for short, implement residual blocks. These blocks allow the gradient to flow easily through the network and skip several layers through an identity shortcut. Inside the block are two or three convolutional layers with ReLU and batch normalization. ResNets can have a depth of up to hundreds of layers (e.g., ResNet-152). ResNets, and more specifically the residual blocks, are some of the most commonly used network architectures today [78].


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

.

Figure B.8: Residual block, taken and edited from [78]

B.8.2 U-Net

U-Net was initially designed in 2015 as a way of segmenting biomedical images. The architecture consists of the contracting path and the expansive path, first reducing spatial information while increasing the information about the features. The latter takes this information and applies it to the entire image on a pixel-by-pixel basis [79].

Contracting path is similar to a standard CNN structure, with each block having two convolutional layers with a 3×3 kernel, doubling the number of channels and 2×2 pooling. The expansive path then expands the features with up-convolutional layers, halving the number of channels from corresponding feature maps found in the contracting path while simultaneously allowing for details lost in the downsampling to be used in the upsampling. Finally, the last layer is a convolution layer with a 1×1 kernel that maps to the number of desired output classes [79].

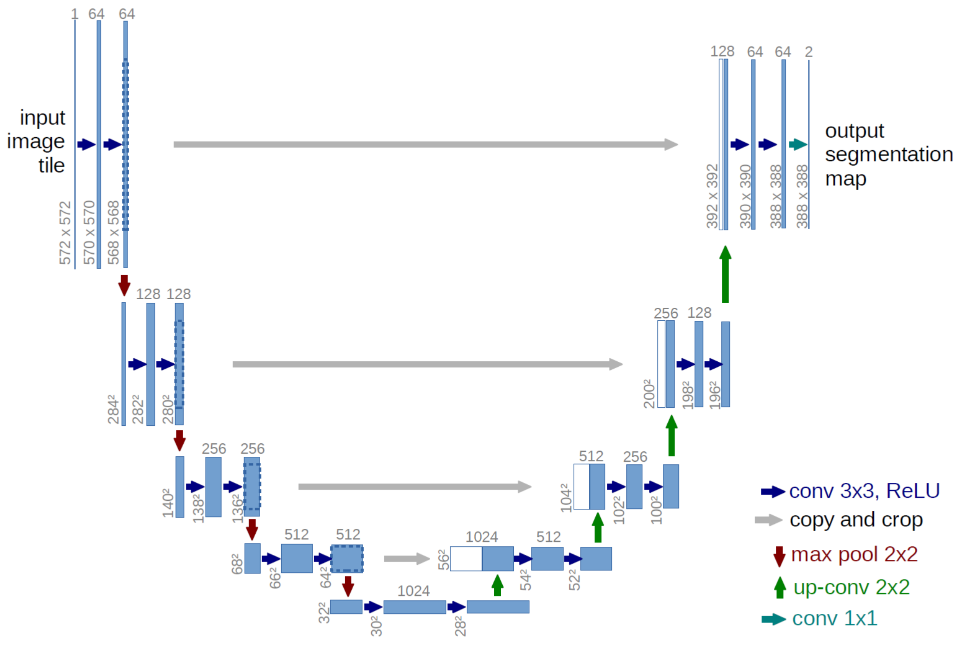


Figure B.9: U-Net architecture, taken and edited from [79]

B.9 Transfer learning

Transfer learning is a technique used in machine learning, which uses learned parameters from different datasets to "boot-start" learning on another dataset [80], effectively reducing the time required for learning. Especially several first layers of the network are affected, as they are usually responsible for feature extraction, which could vary widely between datasets. However, deeper layers are responsible for more general tasks, like the overall movement of the found features, which is usually similar between datasets and is transferable [81].

Appendix C
Implementation

All scripts are available from the GitLab repository: `https://gitlab.fel.cvut.cz/biomedical-imaging-algorithms/deep_registration.git`.

Scripts taken but partially edited from other sources are shown with an asterisk *.

List of required libraries:

- **Numpy** - General matrix operations
- **PyTorch** - Machine learning and neural networks framework (CUDA capable GPU is required)
- **SciPy** - Algorithms for scientific computation
- **PIL** - Image loading and augmentation
- **cv2** - Image augmentation and manipulation
- **scikit-image** - Image augmentation
- **Matplotlib** - Image visualization
- **csv** - Landmark loading

The length of our code is around 3500 lines, not including code taken from other sources. It is split into 21 files.

Not included is the network architecture itself, which is available from the `https://github.com/zacjiang/GMA`.

Structure of the directory

- src
  - data_generation
    - dataset_generation.py
    - downsample_dataset.py
    - generate_affine.py
    - generate_flow_affine_and_synthetic.py
    - generate_flow_landmarks.py
    - mask_creation.py
    - synthetic_flow.py
  - registration
    - metric.py
    - register_course_to_fine.py
    - register_global.py
    - register_local.py
  - training
    - census_loss.py*
    - datasets.py*
    - loss_functions.py
    - sequence_loss.py*
    - smooth_loss.py
    - ssim_loss.py*
    - train.py*
  - utils
    - data_management.py
    - utils_warp.py
    - visualize_data.py

- **data_generation** - Scripts in this folder are responsible for the creation of the training dataset. The user is expected to provide a folder with the images and landmarks with the same structure as the ANHIR dataset. Output training data is then placed into a single folder.

- **registration** - The content of this folder is centered around the different parts of the registration process and result quantification. The input of the different methods is a image pair, alongside with their landmarks, however, only the image can also be used (scripts are written with landmarks in mind, but parts of the scripts can be used for image-only registration).

- **training** - These scripts are responsible for the training of our network, ranging from different loss functions, dataset augmentation, and loading to the training procedure itself. The input must be in the shape of the dataset generated by the scripts in the **data_generation** folder.
- **utils** - Several shorter scripts, or scripts which are commonly used in larger part of the codebase are placed in this folder.