Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Robust and Fast Local All Pass Image Registration

David Kunz

Supervisor: Prof. Dr. Ing. Jan Kybic
August 2020

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kunz David**	Personal ID number:	**467838**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Branch of study:	**Computer and Information Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Robust and Fast Local All Pass Image Registration

Bachelor’s thesis title in Czech:

Robustní a rychlá registrace obrazů metodou "Local All Pass"

Guidelines:

Get familiar with the Local All Pass (LAP) and related image registration methos, test the performance of the LAP method experimentally. Modify the method to efficiently evaluate deformation at a sparse set of points. Evaluate the speed-up. Fit a global deformation model (rigid, affine, B-spline etc.) to this sparse displacement field. Exper-imentally evaluate the performance of this registration algorithm and compare it with alternatives.
Time permitting, implement a method for estimating the deformation prediction uncertainty and use it to improve the global deformation model fit. Evaluate experimentally.

Bibliography / sources:

[1] Gilliam, C. & Blu,https://anhir.grand-challenge.org/ T.,"Local All-Pass Geometric Deformations", IEEE Transactions on Image Processing, Vol. 27 (2), pp. 1010-1025, February 2018
[2] Kybic, J. and Borovec, J.. "Fast registration by boundary sampling and linear programming." MICCAI, vol. 11070, pp. 783-791, 2018
[3] Automatic Non-rigid Histological Image Registration (ANHIR) challenge, https://anhir.grand-challenge.org/
[4] Zitová, B.,Flusser, J. : Image registration methods: a survey. Image and Vision Computing, Volume 21, Issue 11, October 2003, Pages 977-1000

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: **09.01.2020** Deadline for bachelor thesis submission: **14.08.2020**

Assignment valid until: **30.09.2021**

prof. Dr. Ing. Jan Kybic	doc. Ing. Tomáš Svoboda, 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 project supervisor for being patient and supportive of me in this research. I would also like to thank my friends, specifically Bára and Dominik, and my family for supporting me.

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.

Prague, August 14, 2020,

Prohlašuji, že jsem předloženou práci vypracoval samostatntě a že jsem uvedl veškeré použité zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 14. srpna, 2020

Abstract

Image registration is an important part of many practical applications in the field of medical imaging, computer vision, cartography etc. In this bachelor's thesis I will present a new method of registration based on the "Poly-Filter Local All-Pass" (PF-LAP) method. The PF-LAP algorithm uses systems of linear equations to estimate an all-pass filter representing the local displacement of each pixel in a coarse to fine manner. The new proposed method uses a similar approach to finding the local deformation, but does this only for a sparse set of chosen pixels. The sparse deformation field is then fit into a global deformation model. The effectiveness and speed of the new proposed method is experimentally compared with a four other methods including the PF-LAP method. The results show that the proposed method is faster and more accurate on the chosen datasets, than most of the other methods, including PF-LAP.

Keywords: image registration, image processing

Supervisor: Prof. Dr. Ing. Jan Kybic
FEE, Department of Cybernetics

Abstrakt

Registrace obrázků je důležitou součástí mnoha praktických aplikací v poli lékařského zobrazování, v počítačovém vidění, v kartografii atd. V této bakalářské práci představuji novou metodu registrace založenou na metodě "Poly-Filter Local All-Pass" (PF-LAP). Algoritmus PF-LAP používá soustavy lineárních rovnic k odhadu all-pass filtru representující lokální posun každého pixelu iterativně, od hrubé po jemnou deformaci. Nová navrhovaná metoda používá obdobný způsob nalezení lokální deformace, ale hledá ji pouze pro řídkou sadu vybraných pixelů. Získaná řídká deformace je aproximována globálním deformačním modelem. Účinnost a rychlost nové navrhované metody je experimentálně porovnána s dalšími čtyřmi metodami, včetně metody PF-LAP. Výsledky ukazují, že navrhovaná metoda je rychlejší a přesnější na vybraných datasetech, než většina testovaných method včetně PF-LAP.

Klíčová slova: registrace obrázků, zpracovávání obrazu

Překlad názvu: Robustní a rychlá registrace obrazů metodou "Local All Pass"

Contents

Figures

Tables

Chapter 1
Introduction

This thesis focuses on image registration [3, 4, 5]. Image registration is the process of overlaying different images capturing the same scene.

Motivation and Goals

Image registration is essential in many different fields, for different uses. In medical imaging, for example, it is used for combining computer tomography (CT) scans with magnetic resonance (MR) images for more complete information about the patient or for monitoring tumor growth.

There are different approaches to image registration algorithms each with its own uses and its own advantages and disadvantages. I will focus on the Local All-Pass (LAP) and Poly-Filter Local All-Pass (PF-LAP) algorithms [5, 6] developed by Christopher Gilliam and Thierry Blu, with a goal to develop a faster version of the LAP registration method, using a sparse evaluation and global deformation fitting.

Thesis Structure

In Chapter 2, I explain what image registration is and I give an overview of image registration algorithms. In the following chapter I describe in detail how the LAP and PF-LAP algorithms work in particular. Afterwards, in Chapter 4, I present a new image registration method that I developed for this thesis, which is based on the PF-LAP method. Then, in Chapter 5, I evaluate the new method experimentally and compare it to PF-LAP and other image registration methods.

Chapter 2

Image Registration

Image registration is the process of finding a geometric transformation that takes one image, I_2 — known as the *source/moving/sensed* image — and aligns it with another image, I_1 — known as the *target/fixed/reference* image.

The differences between I_1 and I_2 are due to different imaging conditions; these images can be taken at different times, at different depths, from different viewpoints, by different sensors etc. The goal of image registration is to find the geometric transformation mapping these two images. The differences of the images can be classified according to the sensor used to acquire the images: images taken by the same sensor are known as mono-modal images (see Figure 2.2) and images taken by different sensors are known as multi-modal images (see Figure 2.1). Generally speaking, multi-modal image pairs are more different in appearance than mono-modal images.

2.1 Formal Definition

A grayscale image can be thought of as a function that maps a pixel location — a coordinate — to an intensity $I(\mathbf{x}) = e$, where $\mathbf{x} \in \mathbb{R}^N$ is the pixel location, $e \in \mathbb{R}$ is the intensity and N is the dimensionality of the image.

A displacement field is a function that maps a pixel location to a displacement vector $u(\mathbf{x}) = [u_1(\mathbf{x}), \ldots, u_N(\mathbf{x})]^T$, where $\mathbf{x} \in \mathbb{R}^N$ is the pixel location, u_n is the displacement in the n^{th} dimension and N is the dimensionality of the image.

In this thesis I will be dealing only with 2D images, so this notation can be simplified as follows:

$$\begin{gathered} I(x,y) = e, \\ u(x,y) = \left[u_x(x,y), u_y(x,y)\right],^T \\ \text{where: } u_x : \mathbb{R}^2 \to \mathbb{R}, \\ u_y : \mathbb{R}^2 \to \mathbb{R}. \end{gathered} \tag{2.1}$$

Here the the u_x is the shift of the pixel at (x, y) in the x direction and u_y is the shift of the pixel at (x, y) in the y direction.

Two images can be related directly, by a geometric transformation $\mathcal{T}$, if their pixel intensities do not change as they are moved from one image to

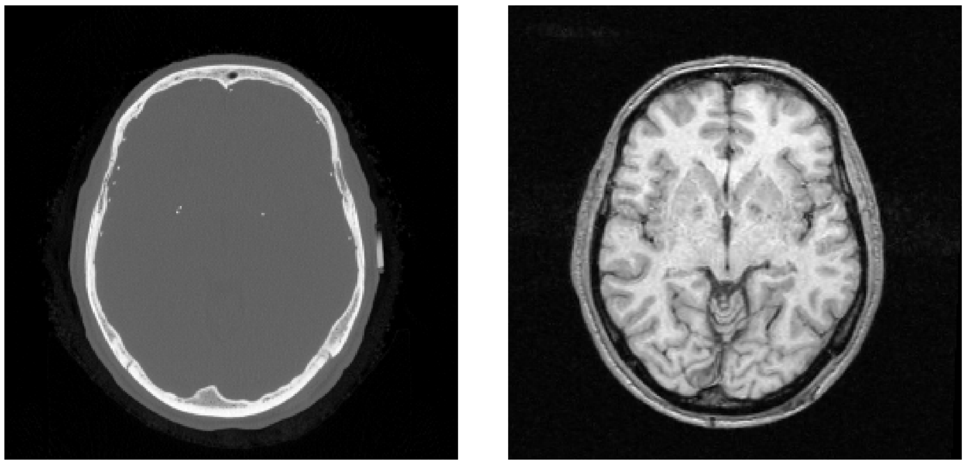


Figure 2.1: multi-modal images taken from [1], a computer tomography (CT) image on the left and a magnetic resonance (MR) image on the right.

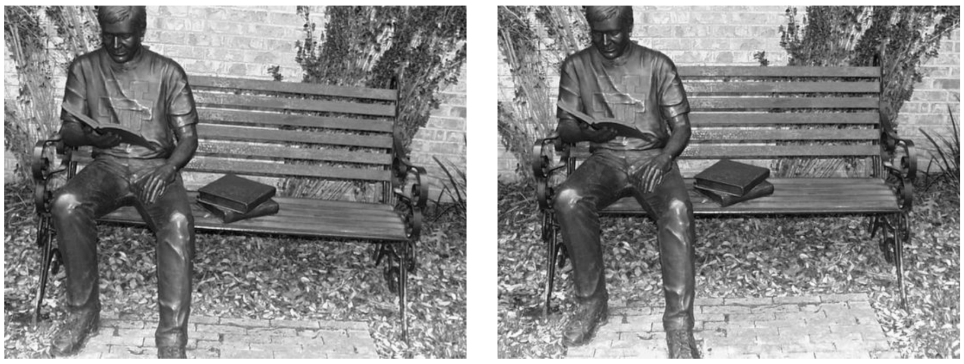


Figure 2.2: mono-modal images taken from [2]

another. In optical flow theory this is known as the **brightness constancy** hypothesis [7].

Under The brightness constancy assumption two images are related as follows:

Definition 2.1. Brightness constancy

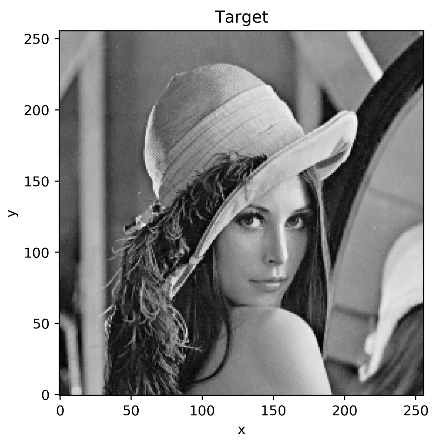
$$\begin{aligned} I_1(x,y) &= I_2(\mathcal{T}(x,y)), \\ \mathcal{T}(x,y) &= (x + u_x(x,y), y + u_y(x,y)) \end{aligned} \tag{2.2}$$

This formulation is, however, very restrictive, it means that the images have to have the same illumination. This is generally not the case for multi-modal (2.1) nor mono-modal images (2.2).

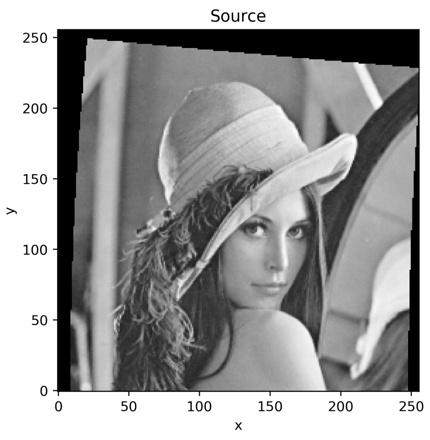
Definition 2.2. Image registration is estimating $u(x,y)$ so that image I_2 is aligned with I_1.

Example 2.3. In Figure 2.3 is an example of how I_1, I_2, $u(x,y)$ from equation 2.1 can look like.

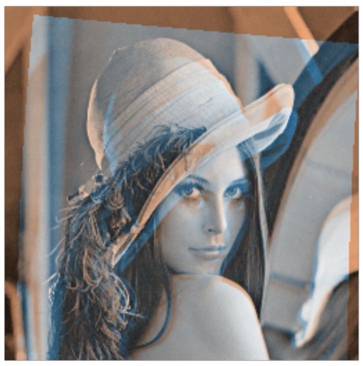
Remark 2.4. So far images, displacement fields, etc., were used in a continuous sense — with independent variables being $x \in \mathbb{R}$ and $y \in \mathbb{R}$ and surrounded



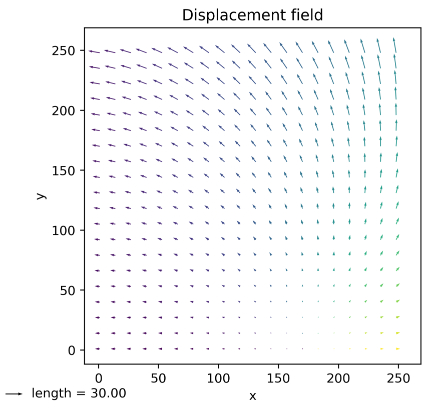
(a) : $I_1(x, y)$



(b) : $I_2(x, y)$



(c) : I_1 (orange) and I_2 (blue)



(d) : $u(x, y)$

Figure 2.3: The Lena photo in 2.3a as the target image, the warped source image in 2.3b, a blending as a representation of differences between the target and source in 2.3c and the ground truth displacement field that transforms the source image I_2 so that it's aligned with I_1 in 2.3d.

by round brackets. Later, when working with functions in the discrete domain, I will use $k \in \mathbb{Z}$ and $l \in \mathbb{Z}$ as independent variables and surround them with square brackets.

2.2 State of the Art

Image registration algorithms can be split roughly into two categories by the nature of their matching criteria (feature-based, area-based) [3], and two categories by the nature of their deformation model (global, elastic).

Feature-based methods are based on extracting significant features that are shared by both images, these can be SIFT points [8] or regions (forests, biological tissue), lines (rivers, region boundaries like; edges of bones, lakes) or points (lines intersections, region corners). Ideally these features are distinct and spread all over the image. Compared to area-based methods, these methods do not work directly with intensity, the features are on a higher information level [3].

Intensity-based methods deal directly with intensities of images without detecting significant points. They use similarity measures like normalized cross-correlation, mutual information, or the sum of squared differences on windows of images to find correspondence.

Parametric methods use global parametric models to describe the deformation field. The deformation field can be represented using basis functions, such as quadratic functions [9] or B-splines [10]. The advantages of these methods are that many deformations can be approximated by using a small number of parameters.

Elastic methods deal with per pixel displacement estimations, they are generally able to estimate more complex displacement fields than parametric methods, but can be more sensitive to intensity changes.

Chapter 3

The Local All-Pass Algorithm

In this chapter, I talk about the method Local All-Pass (LAP) method first introduced in [6] by Christopher Gilliam and Thierry Blu.

The main concept of the Local All-Pass algorithm is that a constant displacement between two images, I_1 and I_2, is equivalent to filtering with an all-pass filter. This comes from time-shifting property of the Fourier transform; $\mathcal{F}\{f(t-t_0)\} = e^{-j\omega t_0}F(\omega)$. Therefore, estimating this constant displacement field boils down to finding an all-pass filter and extracting the displacement from it. I will elaborate on how this is done in the next section using [5] as a reference. This basic concept applies if the displacement field is constant. But what if it isn't? We can make a different assumption about the deformation field; we assume that the displacement is slowly varying, which means it is *locally* constant. More in section 3.2.

3.1 LAP on a Constant Displacement Field

In a constant displacement field $u(x, y)$ all vectors are identical, so $u(x, y) = [a, b]^T$ where a and b are constants.

Example 3.1. In Figure 3.1 is an example of what a constant displacement field (and images shifted with it) can look like.

Shifting is All-Pass Filtering

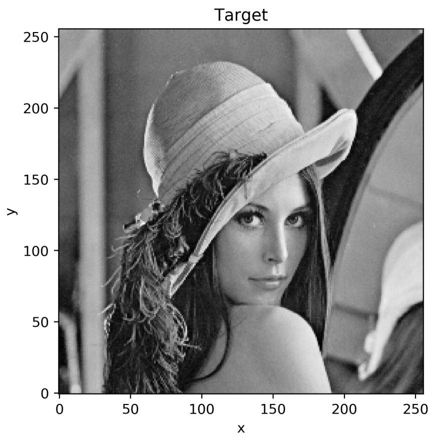
We have two images, I_1 and I_2, related by a constant displacement field $u(x, y)$ (just as in Example 3.1). Assuming brightness constancy 2.1, I_1 is a shifted I_2:

$$I_1(x, y) = I_2(x + u_x(x, y), y + u_y(x, y)). \tag{3.1}$$

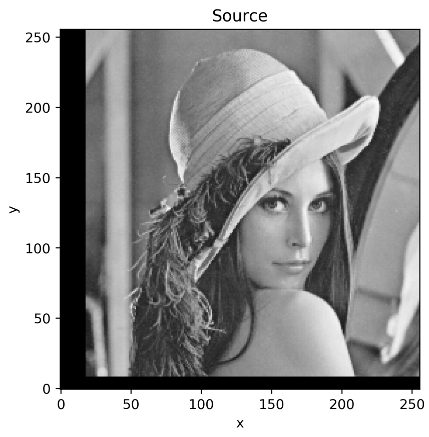
More specifically, since we are working with a constant displacement field

$$I_1(x, y) = I_2(x + a, y + b). \tag{3.2}$$

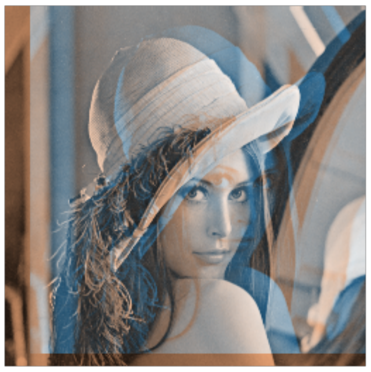
A constant shift in the spatial domain is equivalent to the following in the frequency domain:



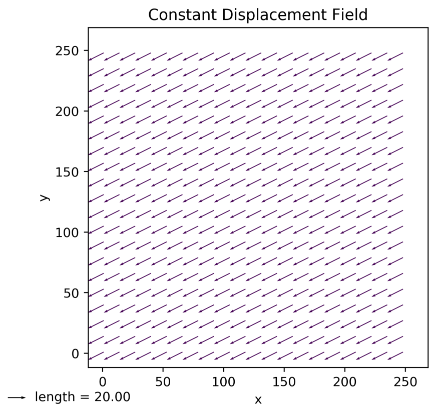
(a) : $I_1(x, y)$



(b) : $I_2(x, y)$



(c) : I_1 (orange) and I_2 (blue)



(d) : $u(x, y)$

Figure 3.1: The target image in 3.1a, is warped by a constant displacement field opposite to the one in 3.1d, the result is the source image 3.1b. In 3.1c there is a blending of these images.

$$\hat{I}_1(\omega_x, \omega_y)e^{-ja\omega_x - jb\omega_x} = \hat{I}_2(\omega_x, \omega_y), \tag{3.3}$$

where $\hat{I}_1$ and $\hat{I}_2$ are Fourier transforms of images I_1 and I_2 and (ω_x, ω_y) are coordinates in the frequency domain. We can define a filter with a frequency response

$$\hat{h}(\omega_x, \omega_y) = e^{-ja\omega_x - jb\omega_x}. \tag{3.4}$$

And we see that I_2 is a filtered version of I_1 with the filter $\hat{h}(\omega_x, \omega_y)$. This filter is,

1. separable: $\hat{h}(\omega) = \hat{h}_1(\omega_1)\,\hat{h}_2(\omega_2)$, where $\hat{h}_1$ and $\hat{h}_2$ are two 1D filters,
2. real: $\hat{h}(\omega) = \hat{h}^*(-\omega)$, where $\hat{h}^*$ represents the complex conjugate of $\hat{h}$, and
3. all-pass: $|\hat{h}(\omega)| = 1$, as shown in [6].

To estimate the filter we need to move to the discrete domain. The properties of the digital version of h are the same as the continuous version.

3.1.1 Estimating the Filter h

The frequency response $\hat{h}(\omega_x, \omega_y)$ of a digital all-pass filter can be expressed as the ratio of the Discrete Fourier transform (DFT) of two filters that have opposite phase [5]. This means that $\hat{h}(\omega_x, \omega_y)$ can be expressed as

$$\hat{h}(\omega_x, \omega_y) = \frac{\hat{p}(e^{j\omega_x}, e^{j\omega_y})}{\hat{p}(e^{-j\omega_x}, e^{-j\omega_y})}, \tag{3.5}$$

where $\hat{p}(e^{j\omega_x}, e^{j\omega_y})$ is called the forward filter and $\hat{p}(e^{-j\omega_x}, e^{-j\omega_y})$ the backward filter.

Using equations 3.3, 3.5 and 3.4, we get

$$\begin{aligned} \hat{I}_1(\omega_x, \omega_y)\hat{h}(\omega_x, \omega_y) &= \hat{I}_2(\omega_x, \omega_y), \\ \hat{I}_1(\omega_x, \omega_y)\hat{p}(e^{j\omega_x}, e^{j\omega_y}) &= \hat{I}_2(\omega_x, \omega_y)\hat{p}(e^{-j\omega_x}, e^{-j\omega_y}). \end{aligned} \tag{3.6}$$

In the discrete spatial domain — using the discrete pixels locations k, l as independent variables as mentioned before in *Remark* 2.4 — the equation 3.6 becomes

$$\begin{aligned} I_1[k,l] * h[k,l] &= I_2[k,l], \\ I_1[k,l] * p[k,l] &= I_2[k,l] * p[-k,-l] \\ &\Updownarrow \\ I_1[k,l] * p[k,l] - I_2[k,l] * p[-k,-l] &= 0. \end{aligned} \tag{3.7}$$

Remark. $*$ denotes convolution. Two dimensional discrete convolution of an image $f[k,l]$ and a filter w can be defined as follows:

$$g[k,l] = w[k,l] * f[k,l] = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w[s,t] f[k-s, l-t]. \tag{3.8}$$

Since a digital image is of finite extent, convolution is undefined at the borders of the image. In particular, for an image $f[k,l]$, of size $M \times N$, $f[k \pm s, l \pm t]$ is only defined for $1 \leq k \pm s \leq N$ and $1 \leq l \pm t \leq M$. This is addressed by artificially expanding the domain of the image by padding. In the LAP algorithms, symmetric padding is used.

Remark. Symmetric padding of the string *abcdef* would look like this:

$$\boxed{d\,c\,b\,a \;\big|\; a\,b\,c\,d\,e\,f \;\big|\; f\,e\,d\,c}$$

Estimating the all-pass filter h is the equivalent to estimating the forward filter p.

Estimating the Forward Filter p. The filter p is then approximated as a linear combination of N known basis filters p_n:

$$p_{\text{approx}}[k,l] = \sum_{n=0}^{N-1} c_n p_n[k,l] \tag{3.9}$$

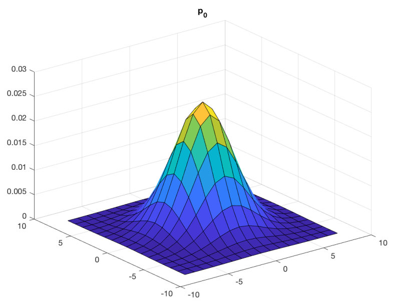
where c_n are the coefficients of the filters p_n.

The basis filters chosen in [5] are:

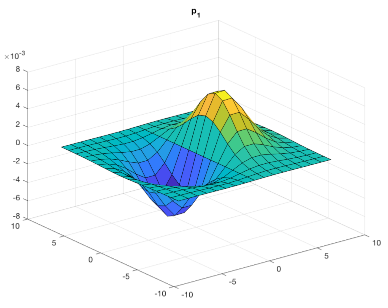
$$\begin{aligned} p_0[k,l] &= \exp\left(-\frac{k^2+l^2}{2\sigma^2}\right), \\ p_1[k,l] &= k\, p_0[k,l], \\ p_2[k,l] &= l\, p_0[k,l], \\ p_3[k,l] &= (k^2+l^2-2\sigma^2)\, p_0[k,l], \\ p_4[k,l] &= kl\, p_0[k,l], \\ p_5[k,l] &= (k^2-l^2)\, p_0[k,l], \end{aligned} \tag{3.10}$$

where $-R \le k \le R$, $-R \le l \le R$, $\sigma = (R+2)/4$ and R is the half size of the filters. This means that each basis filter $p_n[k,l]$ is defined on a matrix $(2R+1) \times (2R+1)$ with offset indices, so that the origin $[0,0]$ is in the center, like so:

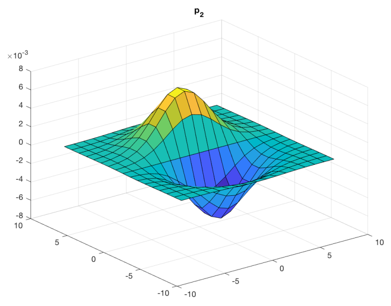
$$\begin{array}{ccccccc}
\boxed{p[-R,-R]} & \cdots & \boxed{p[-R,-1]} & \boxed{p[-R,0]} & \boxed{p[-R,1]} & \cdots & \boxed{p[-R,R]} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \iddots & \vdots \\
\boxed{p[-1,-R]} & \cdots & \boxed{p[-1,-1]} & \boxed{p[-1,0]} & \boxed{p[-1,1]} & \cdots & \boxed{p[-1,R]} \\
\boxed{p[0,-R]} & \cdots & \boxed{p[0,-1]} & \boxed{p[0,0]} & \boxed{p[0,1]} & \cdots & \boxed{p[0,R]} \\
\boxed{p[1,-R]} & \cdots & \boxed{p[1,-1]} & \boxed{p[1,0]} & \boxed{p[1,1]} & \cdots & \boxed{p[1,R]} \\
\vdots & \iddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\boxed{p[R,-R]} & \cdots & \boxed{p[R,-1]} & \boxed{p[R,0]} & \boxed{p[R,1]} & \cdots & \boxed{p[R,R]}
\end{array}$$



(a) : Basis filter $p_0[k,l]$.



(b) : Basis filter $p_1[k,l]$.



(c) : Basis filter $p_2[k,l]$.

Figure 3.2: First 3 basis filters—$p_0[k,l]$, $p_1[k,l]$ and $p_2[k,l]$—with half size $R = 8$.

3.1.2 Retrieving the Deformation from Filters

The frequency response of the estimated filter h_{est} should be close to that of 3.4, so the displacement estimation is

$$u_x, u_y = j \frac{\partial \log\left(h_{\text{est}}(\omega_x, \omega_y)\right)}{\partial \omega_{x,y}}\bigg|_{\omega_x=\omega_y=0}. \tag{3.11}$$

In terms of impulse response of the filter p:

$$\begin{aligned} u_x &= 2\frac{\sum_{k,l} k\, p_{\text{approx}}[k,l]}{\sum_{k,l} p_{\text{approx}}[k,l]}, \\ u_y &= 2\frac{\sum_{k,l} l\, p_{\text{approx}}[k,l]}{\sum_{k,l} p_{\text{approx}}[k,l]}. \end{aligned} \tag{3.12}$$

Remark. Note that result of the sum $\sum_{k,l} p_n[k,l]$ (from equation 3.12) for each basis filter is known beforehand—being either 0 or 1 for the filter basis 3.10.

3.2 LAP on a Smooth, Slowly Varying Displacement Field

In the previous section, we used all the pixels from both images to estimate a single all-pass filter and from it we calculated a single displacement vector that represented the entire *constant* displacement field. Now we will adapt the previous steps to work locally to estimate a smooth, slowly varying displacement field.

In a smooth slowly varying displacement field, the deformation can be considered *locally* constant and so an all-pass filter can be estimated relating a local region in I_1 to the same region in I_2 as illustrated in Figure 3.3.

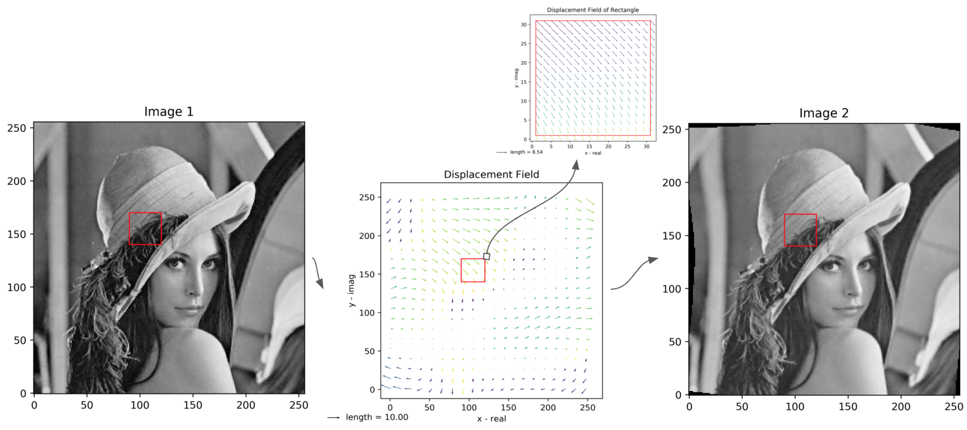


Figure 3.3: A smooth slowly varying displacement field warping Image 1 to Image 2. The red rectangle shows that the displacement can be assumed to be locally constant.

3.2.1 Estimating Local All-Pass Filters

In the Section 3.1, we used all the pixels from both images to estimate a single all-pass filter. Now we will formulate a local version to estimate the filters. There are two key differences compared to the previous section:

1. We are estimating a unique all-pass filter for every pixel g of the image I_2.
2. To estimate the filter for pixel g, we use just some pixels from the images — pixels that are in a square window $\mathcal{W}_g$ around pixel g.

The size of the local window $\mathcal{W}_g$ is $(2W+1) \times (2W+1)$, where $R \leq W$ and it is the size of the location we are assuming to have a constant displacement. In Figure 3.3, there is a visualisation of what this means.

To estimate the displacement of the pixel g, equation 3.7 is modified to be local:

$$I_1[k,l] * p_{\text{approx}}[k,l] = I_2[k,l] * p_{\text{approx}}[-k,-l]$$
$$\Updownarrow$$
$$\sum_{s=-R}^{R}\sum_{t=-R}^{R} p_{\text{approx}}[s,t] I_1[k-s,l-t] = \sum_{s=-R}^{R}\sum_{t=-R}^{R} p_{\text{approx}}[-s,-t] I_2[k-s,l-t], \tag{3.13}$$

where $[k,l] \in \mathcal{W}_g$ and $\mathcal{W}_g$ is the window of pixels with g in the center.

Estimating p_{approx}.

$$\min_{\{c_n\}} \sum_{[k,l]\,\in\,\mathcal{W}_g} \left| p_{\text{approx}}[k,l] * I_1[k,l] - p_{\text{approx}}[-k,-l] * I_2[k,l] \right|^2 ,$$
$$\text{where} \quad p_{\text{approx}}[k,l] = p_0[k,l] + \sum_{n=1}^{N-1} c_n p_n[k,l] \tag{3.14}$$

is minimized to obtain an approximation of the local filter p for pixel g. This is then done at each pixel by shifting the local window $\mathcal{W}_g$, so that it centers around it, obtaining a local filter for each of these pixels. The displacement is then extracted from each of these filters using equation 3.12, making a dense displacement field for the whole image.

Implementation

In this section, I will explain how 3.14 is minimized effectively, then I summarize the entire process in Algorithm 1. Firstly, p_{approx} is substituted into the minimization:

$$\min_{\{c_n\}} \sum_{[k,l]\,\in\,\mathcal{W}_g} |p_0[k,l] * I_1[k,l] - p_0[-k,-l] * I_2[k,l] + \sum_{n=1}^{N-1} \left(c_n p_n[k,l] * I_1[k,l] - p_n[-k,-l] * I_2[k,l] \right) |^2 \tag{3.15}$$

This is simplified by getting rid of the absolute value, then a derivative with respect to c_n is calculated and the result is set to equal 0. For readability, I define the element-wise difference of image $I_1[k,l]$ convolved with $p_n[k,l]$ and image $I_2[k,l]$ convolved with $p_n[-k,-l]$,

$$\psi_n[k,l] \overset{\text{def}}{=} \left(p_n[k,l] * I_1[k,l] - p_n[-k,-l] * I_2[k,l] \right) ,$$

and the sum of element-wise products of two functions $\zeta[k,l]$ and $\xi[k,l]$ in the window $\mathcal{W}_g$,

$$\langle \zeta \xi \rangle_{\mathcal{W}_g} \overset{\text{def}}{=} \sum_{[k,l]\,\in\,\mathcal{W}_g} \xi[k,l]\zeta[k,l],$$

The minimization can then be formulated as follows,

$$0 = \sum_{[k,l]\,\in\,\mathcal{W}_g} \psi_0[k,l]\psi_n[k,l] + \sum_{m=1}^{N-1} c_n \sum_{[k,l]\,\in\,\mathcal{W}_g} \psi_n[k,l]\psi_m[k,l] \tag{3.16}$$
$$\text{for: } n = 1, 2, \ldots, N-1.$$

Therefore, solving 3.14 is equivalent to solving $N-1$ linear equations, which is done as follows:

1. The convolution of image $I_1[k,l]$ with every forward filter $p_n[k,l]$ is calculated and so is the convolution of image $I_2[k,l]$ with every backward filter $p_n[-k,-l]$. That is $2N$ convolutions — N convolutions per image.

2. Calculate $\psi_n[k,l]$ for every n.

3. For every pixel g in the image $I_1[k,l]$ a system of linear equations is prepared as follows:

$$\mathbf{A}_g \mathbf{c}_g = \mathbf{b}_g$$

$$\begin{bmatrix} & & \\ & \mathbf{A}_g & \\ & & \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} \\ \mathbf{b}_g \\ \\ \end{bmatrix},$$

where $\mathbf{A}_g$ and $\mathbf{b}_g$ are constructed like this:

$$\mathbf{A}_g = \begin{bmatrix} \langle \psi_1 \psi_1 \rangle_{\mathcal{W}_g} & \cdots & \langle \psi_1 \psi_{N-1} \rangle_{\mathcal{W}_g} \\ \vdots & \ddots & \vdots \\ \langle \psi_{N-1} \psi_1 \rangle_{\mathcal{W}_g} & \cdots & \langle \psi_{N-1} \psi_{N-1} \rangle_{\mathcal{W}_g} \end{bmatrix}$$

$$\mathbf{b}_g = - \begin{bmatrix} \langle \psi_1 \psi_0 \rangle_{\mathcal{W}_g} \\ \vdots \\ \langle \psi_{N-1} \psi_0 \rangle_{\mathcal{W}_g} \end{bmatrix},$$

and $\mathbf{c}_g$ is a vector of coefficient c_n and the window $\mathcal{W}_g$ is centered around g. The sums over the window $\mathcal{W}_g$ can be calculated quite effectively using a summed-area table (see the following paragraph) or by using convolution.

4. Solve the system of equations for every pixel g obtaining the coefficients vector $\mathbf{c}_g$. This step can be done very effectively using Gaussian elimination by making use of vectorized, elementwise operations if the matrices $\mathbf{A}_g$ are organised is a specific way.

Using a summed-area table. For every pixel g from $I_1[k,l]$ the sum $\langle \psi_n \psi_m \rangle_{\mathcal{W}_g}$ has to be calculated $\frac{(N-1)N}{2} + N - 1$ times — due to the dimensions of $\mathbf{A}_g$ and $\mathbf{b}_g$. But since the window $\mathcal{W}_g$ is centered around each pixel in turn (it is shifted by 1 pixel distances), the sums can be effectively calculated using a summed-area table. This is very effective when we need the calucation done for every pixel g from I_2. To create a summed-area table, for a function $Q[k,l]$, to calculate $\sum_{[k,l] \in \mathcal{W}_g} Q[k,l]$ with $\mathcal{W}_g$ around every pixel of $Q[k,l]$ in turn we follow these steps:

1. $Q[k,l]$ is padded with a zero padding with width W in each direction. This is so that the sum would be defined for pixels within W in of the image edge.

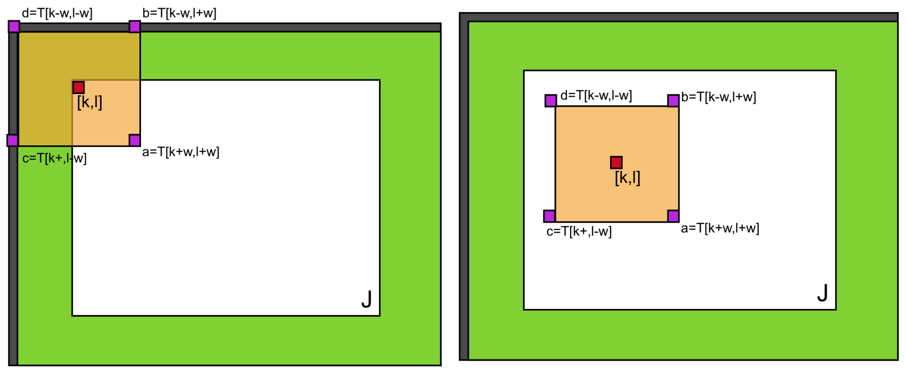
2. The result is padded once more, this time with 1 pixel width of zeros. This is so that the calculation, as shown in Figure 3.4, doesn't have to check for edge conditions.

3. Perform a 2D cumulative sum over the padded image.

Then the sum of the window $\mathcal{W}_g$ around pixel g of function $Q[k,l]$ is just 3 operations on the summed-area table, like in Figure 3.4.

2D cumulative sum. A table T, with a 2D cumulative sum of image I, is created as follows:

$$\begin{aligned} T[0,0] &= 0, \\ T[n+1,n] &= T[n,n] + I[n+1,n], \\ T[n,n+1] &= T[n,n] + I[n,n+1]. \end{aligned} \tag{3.17}$$



(a) : Summed-area table edge case. **(b) :** Summed-area table.

Figure 3.4: A summed-area table for the image $Q[k,l]$. The images show the area of the original image $Q[k,l]$ in white, the symmetric padding of width W in green, the 1 pixel zero padding in black, the area of the window $\mathcal{W}$ in orange and the central pixel of $\mathcal{W}$ in red. The sum $\sum_{[k,l]\in\mathcal{W}} Q[k,l]$ is calculated from the values at the locations marked in purple, with 3 operations: $a-c-b+d$.

Algorithm 1: Local All-Pass (LAP) Estimation of a Deformation Field from [5].

```
  Input : Images $I_1$ and $I_2$, window half size $W$, filter half size $R$ and
          number of filters $N$
1 Initialization: Generate $N$ basis filters, based on $N$ and $R$ (see 3.1.1).
2 Filtering: Filter $I_1$ and $I_2$ with the basis filters and calculate $\psi_n[k,l]$
    for every $n = 1, 2, ..., N-1$.
3 Systems of Equations - Preparation: Prepare matrices $A_g$ and vector
    $b_g$ for every pixel $g$ in $I_2$, using summed-area tables.
4 Systems of Equations - Solution: Solve the linear system $A_g c_g = b_g$ for
    every pixel $g$ in $I_2$.
5 Extraction: Using the calculated coefficients vector $c_g$ calculate the
    deformation for every pixel $g$ in $I_2$, using 3.12
  Result: Deformation field aligning $I_2$ to $I_1$.
```

3.3 Poly-Filter LAP

When the LAP algorithm estimates large deformations it has to use a large window $\mathcal{W}$ and thus has to assume, that large parts of the displacement field are constant. To remedy this, the Poly-Filter LAP (PF-LAP) [5] algorithm uses the LAP algorithm to estimate the displacement field iteratively, in a coarse-to-fine manner. The algorithm starts by estimating the displacement field using a large filter size R allowing for slowly varying deformations and reducing R at each iteration, allowing for faster varying deformations. At each iteration, the deformation is estimated, image I_2 is warped closer to image I_1 using the current deformation estimate and this estimate is added to the previous estimate. The resulting displacement estimate is a sum of the estimates at all the values of R. The specifics of this process are delineated in the flowing sections and are summarized in Algorithm 2.

The estimate of the deformation field yielded from the LAP Algorithm 1 isn't perfect and obvious errors and outlying estimates have to be removed in the each iteration of PF-LAP in a post-processing step. Also, it can be profitable to repeat iterations with one filter size R. To do so, at the end of each iteration the Peak signal-to-noise ratio (PSNR) of the warped source image I_2^{warped} and the target image I_1 is compared to the PSNR of the previous I_2^{warped} and I_1. If the value after, minus before the iteration, is higher than a threshold ϵ, no iteration is added, but if otherwise, an additional iteration with the same value of R is added. New iterations are added only to a certain maximum number of iterations, $Max.Iterations = 3$.

Remark. As one of the initialization steps to improve robustness, a limit is set on the the minimum size of W based on the how much noise is estimated to be in the images. See the paper [5] for the specific calculation.

3.3.1 Post-Processing

The post-processing of the deformation field has two steps, namely inpainting and smoothing, described below.

Inpainting

The deformation field estimated by Algorithm 1 has obvious error of 2 types:

1. displacement vectors that exceed the expected support of the filter — i.e. vectors which have a length larger than R,
2. displacement vectors that are within W from the edge of the image.

The erroneous values of the first type are removed and replaced using an inpainting procedure. The inpainting algorithm from [11] is used, which iteratively replaces the erroneous values with a mean of values from all of non-erroneous neighbors, until they are all re-estimated. The erroneous values near the boundary are replaced by the nearest valid value.

Smoothing

Since we are assuming that the displacement field is slowly varying, it can be smoothed using a Gaussian filter to eliminate any errors that haven't been replaced in the inpainting procedure. The smoothing Gaussian filter has $\sigma = 2W$ and size $(4W + 1) \times (4W + 1)$.

3.3.2 Pre-Processing

To reduce the reliance on the brightness constancy hypothesis and improve its performance on real images which often have changes in illumination [5] employs 2 optional steps: histogram matching [12] and high-pass filtering. The high-pass image I^{hp} of image I is obtained by subtracting the image smoothed with a Gaussian filter I_{smoothed} from the original I.

Algorithm 2: Poly-Filter Extension of the LAP Algorithm from [5].

```
Input : Images $I_1$ and $I_2$, number of filters $N$, an array of filter sizes r
        and maximum number of repetitions at each filter size
        $Max.repeats$.
1  Initialization: Set the starting estimate of the deformation field $u_0 = \mathbf{0}$,
     the image $I_2$ warped by this deformation field $I_2^{\mathrm{warped}} = I_2$ and the
     minimal size of the window $W_{\mathrm{limit}}$ is set based on the estimated noise.
2  for $i$ in Number of filter sizes $r$ do
3      Set LAP parameters: $R = \mathrm{r}[i]$ and $W = \max(R, W_{limit})$.
4      [Optional] Pre-filtering: High-pass image $I_1$ (see 3.3.2).
5      for $j$ in $Max.$ $repeats$ do
6          [Optional] Pre-filtering: High-pass image $I_2^{\mathrm{warped}}$ (see 3.3.2).
7          Displacement Estimation: With the current $N$, $R$ and $W$
             calculate the deformation field $\Delta u$ between $I_1$ and $I_2^{\mathrm{warped}}$
             using the Algorithm 1.
8          Post-Processing: Remove errors from $\Delta u$ using inpainting and
             smoothing with a Gaussian filter (see 3.3.1).
9          Update the Displacement Estimation: Set $u_i = u_{i-1} + \Delta u$.
10         Warp: Warp $I_2$ with $u_i$ to obtain a new $I_2^{\mathrm{warped}}$.
11         if $\mathrm{PSNR}\left(I_1, I_{2,j}^{\mathrm{warped}}\right) - \mathrm{PSNR}\left(I_1, I_{2,j-1}^{\mathrm{warped}}\right) > \epsilon$ then
12             Break inner loop.
13         end
14     end
15 end
Result: Deformation field aligning $I_2$ to $I_1$ and the registered source
        image, $I_2^{\mathrm{warped}}$.
```

Chapter 4
The Sparse Local All-Pass Algorithms

In this chapter, I will present the new image registration methods, I developed for this thesis, Sparse Local All-Pass (Sparse LAP) and Sparse Poly-Filter Local All-Pass (Sparse PF-LAP). The key idea behind these methods, is to obtain displacement vectors, calculated in the same fashion as the in the LAP algorithm, but only at specific points in the image (not for every pixel of the source image). And then fit a global deformation model to these this sparse set of vectors and thus obtain a dense displacement field.

This approach leads to a few advantages. Speed is improved, because the post-processing and pre-filtering steps can be omitted (see Section 4.5). Also, with global fitting, the deformation field is smoother—naturally does not have outlying estimates.

In the following sections I will explain how this proposed method works and, to describe some of the workings, I will compare it with the LAP and PF-LAP methods.

4.1 Point Selection

Similarly to [4], I assume that correspondences of two images can most easily be found on the boundaries of regions and not as easily in their interiors. This means that pixels that are on the edges or corners of the image features, can provide more reliable information about the displacement in that area. A demonstration of this can be seen in Section 4.1.1.

Points on edges and corners of a image I are selected by smoothing I with a small Gaussian filter with $\sigma = 1$ (to get rid of pixel-sized edges). Then, I is filtered with a Sobel filter [13] in both x and y directions, giving us the gradient in these directions. Then the magnitude of these gradients is calculated, resulting in an edge image (see Figure 4.2). The pixels with the highest intensity are then picked iteratively from this edge image, while the pixels inside a given radius C, of the picked pixel, are restricted from being picked. This, along with providing a maximal number of chosen points D, allows for the points to be sparsely distributed throughout the whole image.

This process is shown in Algorithm 3.

Algorithm 3: Find Edge Points.

```
Input : Image $I$, maximal number of points $D$ and the minimal
        exclusion radius $C$.
1  Gradient Images: Smooth the image $I$ and filter with a Sobel filter in
     both $x$ and $y$ directions obtaining $I^{G_x}$ and $I^{G_y}$ respectively.
2  Edge Image: Calculate the edge image as the magnitude of gradient
     images $I^{edge} = \sqrt{{I^{G_x}}^2 + {I^{G_y}}^2}$.
3  Sort Edge Image Pixels: Sort the pixels of $I^{edge}$ by their intensity and
     store it in $A_{order}$.
4  Initialization: Create an empty starting array of edge points $P$ and a
     empty matrix for marking excluded pixels $E$.
5  for $a_i$ in $A_{order}$ do
6      if $a_i$ not marked as excluded in $E$ then
7          Add $a_i$ to the array $P$.
8          Mark pixels in a radius $C$ from $a_i$ as excluded in $E$.
9          if Number of elements in $P$ $= D$ then
10             Break.
11         end
12     end
13 end
Result: Sparse set of pixels $P$.
```

4.1.1 Edge Image Displacement Estimation Quality

To show that the displacement estimation is better at points of high gradient, I took 100 random images images from the AAPM RT-MAC Grand Challenge 2019 dataset [14, 15] containing MR scans of the head and neck (described in better detail in Chapter 5, with example images in 5.2) and warped them with a random uniform deformation with maximum displacement of 5 pixels. Then I used the LAP algorithm to estimate the displacement field for these images and, knowing the ground truth displacement field, I calculated the normalized cross correlation (NCC) between the displacement error of the estimation and the intensity of the magnitude of gradient of the source. The result of this experiment is shown as a histogram of the calculated NCC values in Figure 4.1.

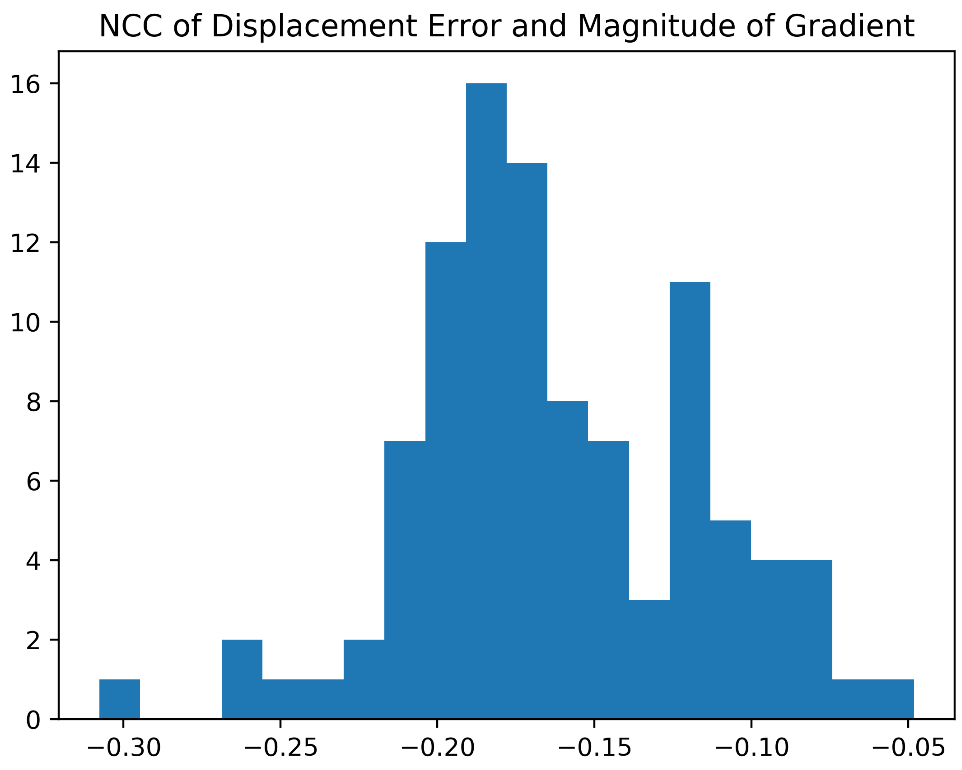


Figure 4.1: Histogram of NCC of displacement error and intensity of the edge image, calculated on 100 random realisations.

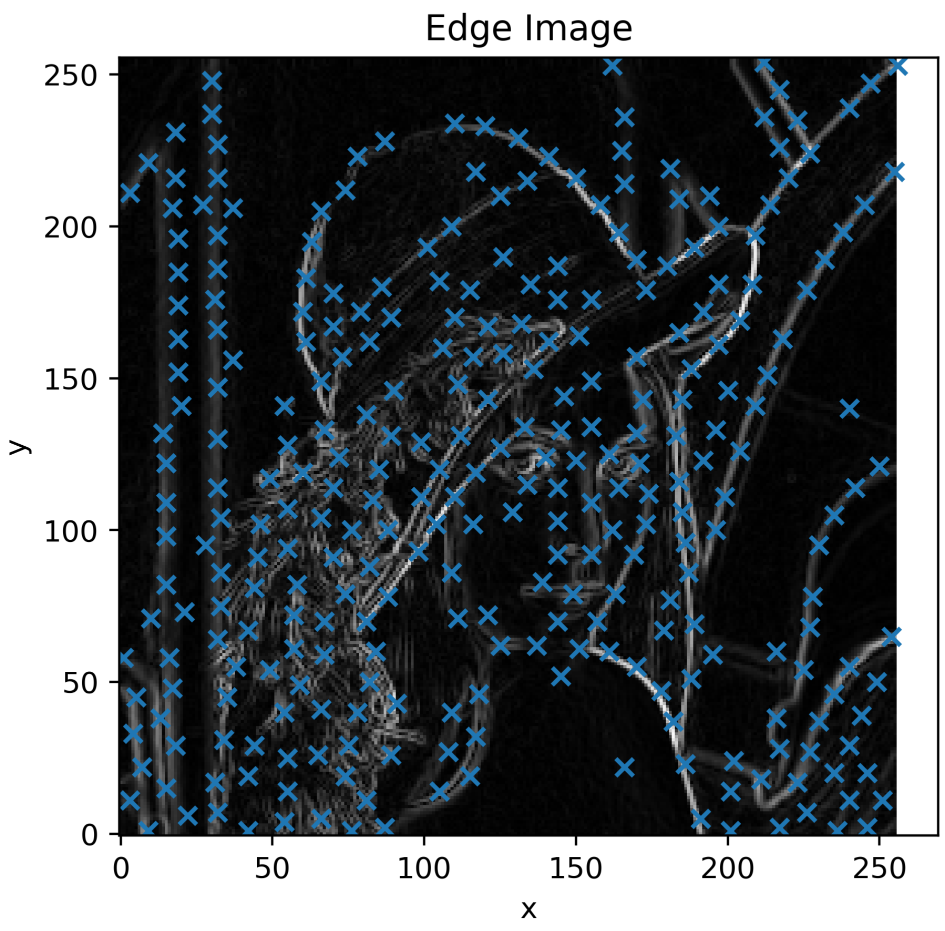


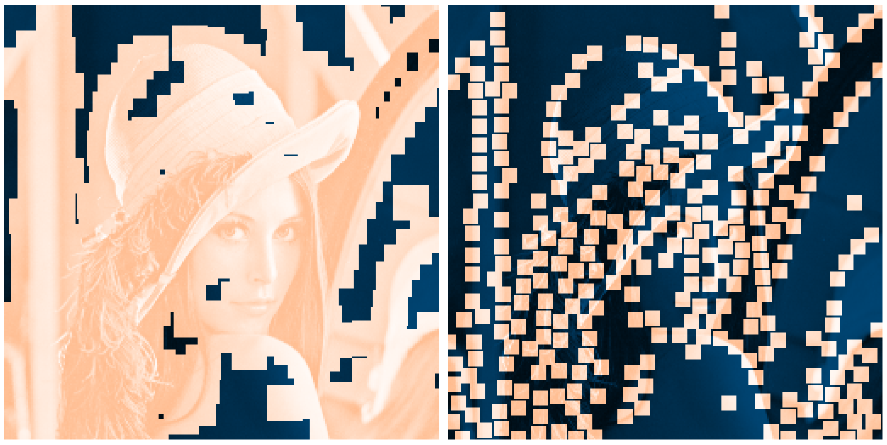
Figure 4.2: Edge image of Lena, with points selected using Algorithm 3.

4.2 Sparse Deformation Field

After selecting the points of interest, a deformation vector for each of these points in calculated. Fortuitously, one of the properties of the LAP algorithm is that estimating an all-pass filter for one pixel is independent of the the estimation for any other pixel — in the way that the minimization (3.14) is separate for each of these pixels. In my work, I take advantage of this and estimate all-pass filters only for some pixels P obtaining a sparse set displacement vectors. To do so, the same steps as in Section 3.2.1 are followed — but only for these specific pixels.

4.2.1 What is different?

Convolved area. In contrast to the implementation in Section 3.2.1, in general, we don't need to perform full convolutions of I_1 and I_2 with the base filters p_n. We only need to convolve I_1 and I_2 with the base filters p_n in the widow $\mathcal{W}_g$ around each pixel g from P.



(a) : $W = R = 10$ with approximately 85% of the image covered.

(b) : $W = R = 4$ with approximately 25% of the image covered.

Figure 4.3: Two Lena images, covered with the windows $\mathcal{W}_g$ around pixels $g \in P$ highlighted in orange.

This means that, in theory, there are some computations to be saved, because only a fraction of the convolutions need to be calculated. However, when collecting the sparse set of points, one of the goals is to have these points well distributed throughout the image. And when estimating the displacement of maximum size O, we need the convolutions of an area of size $R \geq O$, and for medium and large values of R, where the most time would be saved, most of the image is covered with regions, that need a convolution. This is shown in the Figure 4.3a, where $W = R = 10$ and 300 points are chosen from the Lena image, where the area needed to be convolved is in shown in orange. This still leaves some places where nothing has to be calculated, but thanks to the fact that image convolution is very well optimised, when calculated on the whole image, doing the convolution only for the regions in marked in orange is significantly slower. This is true until about 20% of the image is covered, where the difference evens out. But with a healthy amount of points, this 20% coverage should be reached only by the smallest filters with $R < 4$.

Systems of equations. Another possible change to the implementation in Section 3.2.1 is the way the systems of linear equations are solved. Besides Gaussian elimination, I considered QR and Cholesky decompositions, but for $N = 3$ and $N = 6$, ($\mathbf{A}_g$ and $\mathbf{b}_g$ have sizes $(N-1) \times (N-1)$ and $(N-1)$ respectively (see section 3.2.1) and any number of points D (there are up to D systems of equations—one for each chosen pixel), vectorized Gaussian elimination, that can solve all of the equations at once, was always the fastest.

4.3 Global Deformation Model

In [2] the results of the PF-LAP algorithm at each iteration are fit into a global deformation field. According to the paper, this led to speed and precision improvement over the PF-LAP algorithm from [5]. I take this a step further, in that I use the LAP algorithm to calculate the displacement vector only for some pixels, and I fit this this sparse estimate into a global deformation model.

This can be done in a couple of ways, such as affine, quadratic, radial basis function (RBF) or B-spline fitting. In this work, I implemented two options, radial basis function (RBF) fitting and quadratic polynomial fitting.

4.3.1 Radial Basis Function Fitting

Radial basis function fitting allows for good flexibility in the variety of displacement fields. They can create less rigid and more general displacement fields compared to a quadratic polynomial fitting.

Generating a Dense Displacement Field

The dense displacement field $u(x, y)$ is calculated as follows,

$$u(x,y) = \sum_{i=1}^{D} w_i \phi\left(\|(x,y) - (x_i,y_i)\|\right), \tag{4.1}$$

where $\phi(r)$ is the radial basis function, (x_i, y_i) is the coordinate of computed displacement vector $u(x_i, y_i)$ and D is the number of computed displacement vectors. I considered two radial basis functions, multiquadratic,

$$\phi_1(r) = \sqrt{1 + (\varepsilon r)^2},$$

where $\varepsilon = 1$ and thin plate spline [16],

$$\phi_2(r) = r^2 \ln(r).$$

The weights w_i are obtained by solving the matrix equation.

$$\begin{bmatrix} \phi\left(\|(x_1,y_1) - (x_1,y_1)\|\right) & \dots & \phi\left(\|(x_D,y_D) - (x_1,y_1)\|\right) \\ \vdots & \ddots & \vdots \\ \phi\left(\|(x_1,y_1) - (x_D,y_D)\|\right) & \dots & \phi\left(\|(x_D,y_D) - (x_D,y_D)\|\right) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_D \end{bmatrix} = \begin{bmatrix} u_{\text{est}}(x_1,y_1) \\ \vdots \\ u_{\text{est}}(x_D,y_D) \end{bmatrix} \tag{4.2}$$

where $u_{\text{est}}(x_i, y_i)$ is the estimated displacement vector at (x_i, y_i).

4.3.2 Quadratic Polynomial Fitting

In comparison to the displacement field from RBF fitting, quadratic fitting can create much simpler deformations, but when the number of points used is high, it is a lot faster.

Generating a Dense Displacement Field

The dense displacement field $u(x, y)$ is calculated as follows,

$$u(x, y) = \sum_{m=1}^{M} b_m u_m(x, y) \tag{4.3}$$

where $M = 6$ is the number of basis functions u_m, that form the quadratic polynomial. The basis functions are:

$$\begin{aligned} u_1(x, y) &= 1, \\ u_2(x, y) &= x, \\ u_3(x, y) &= y, \\ u_4(x, y) &= x^2, \\ u_5(x, y) &= y^2, \\ u_6(x, y) &= xy. \end{aligned} \tag{4.4}$$

The weights b_m are obtained by minimizing the the square difference of the displacement estimated at coordinate of points P and the quadratic expression 4.3,

$$\min_{b_m} \sum_{x,y \in \mathcal{P}} |u(x, y) - u_{\text{est}}(x, y)|^2 , \tag{4.5}$$

where $\mathcal{P}$ are the coordinates of pixels from P and $u_{\text{est}}(x, y)$ is the estimated displacement vector at (x, y).

With just the first three basis functions u_k, any rigid/linear transformation can be modeled. Adding the next three allows the fitting to deal with more complex situations like perspective changes. In the experiments in Chapter 5, quadratic polynomial fitting was shown to be very effective at approximating homography transformations.

4.4 The Algorithms

Similarly to the previous sections, I first present an algorithm that estimates displacement without iterations — with one filter size R and W — the Sparse Local-All Pass algorithm. And then, I will extend this algorithm, in the poly-filter fashion, so that it is able to estimate faster varying displacement fields.

4.4.1 Sparse LAP

The steps in this algorithm are very similar the to the steps explained in Section 3.2.1, the only difference is that the algorithm takes the edge points P as an extra input parameter, and returns a displacement only for these points. The process is summarized in Algorithm 4.

Algorithm 4: Sparse Local All-Pass Displacement Vector Estimation.

```
Input : Images $I_1$ and $I_2$, number of filters $N$, filter half size $R$,
        window half size $W$ and edge points $P$.
1 Initialization: Generate the filter basis based on $N$ and $R$ (see 3.1.1).
2 Filtering: Filter $I_1$ and $I_2$ with the basis filters and calculate $\psi_n[k,l]$
    for every $n = 1, 2, ..., N-1$.
3 Systems of Equations - Preparation: Prepare matrices $A_g$ and vector
    $b_g$ for every pixel $g$ in $P$, using summed-area tables.
4 Systems of Equations - Solution: Solve the linear system $A_g c_g = b_g$ for
    every pixel $g$ in $P$.
5 Extraction: Using the calculated coefficients vector $c_g$ calculate the
    deformation for every pixel $g$ in $P$, using 3.12
Result: Deformation vectors at points $P$.
```

4.4.2 Sparse PF-LAP

The Sparse Poly-Filter Local All-Pass algorithm (Sparse PF-LAP) is to the Sparse LAP like the PF-LAP is to LAP. The motivation behind it, is to be able to estimate faster varying displacements, using a filter pyramid.

The Sparse PF-LAP algorithm first finds up to D edge points P in the target image using Algorithm 3, then, as in PF-LAP, it estimates the displacement, but this time using Sparse LAP (see Algorithm 4), at a subset of P, P_i, which contains only points that are not within W of the image border (explained in the following paragraphs). Thus obtaining displacement vectors $\mathbf{u}_i$. Then, if this is not the first iteration, deformation vectors at points P_i from the previous dense deformation estimate u_{i-1}, are added to the newly obtained vectors $\mathbf{u}_i$. The newly estimated vectors, added to the vectors taken from the previous global fitting, are now fit into a global deformation model, to obtain the next dense displacement field. After that, the source image is warped closer to the target image, with this dense displacement field. (Optionally, the same PSNR testing mechanism as in PF-LAP is employed, to evaluate if the current registration led to a large improvement.) And then, the next iteration, with a smaller R, is run with the warped source image and target image as inputs.

In contrast to the PF-LAP algorithm, no post-processing is necessary. This plays a key role in the speed advantage of Sparse PF-LAP over PF-LAP (see section 4.5). As for pre-processing, Sparse PF-LAP optionally uses histogram matching before the registration begins.

Subset of P. Similarly to the post-processing step of PF-LAP, where the estimations within W of the image border is scrapped, here, a subset of points P, P_i is created at each iteration, containing only points not in a W boundary of the image border.

Algorithm 5: Poly-Filter Extension of the LAP Algorithm from [5].

```
   Input : Images $I_1$ and $I_2$, number of filters $N$, maximum number of
           points edge points $D$, edge point exclusion radius $C$ and
           maximum number of repetitions at each filter size
           $Max.repeats$.
1  Find Edge Points: Find up to $D$ edge points $P$, with an exclusion
     radius $C$, in $I_1$, using Algorithm 3.
2  [Optional] Histogram Matching: Edit $I_2$ to match the intensity
     histogram of $I_1$ [12].
3  Initialization: Set the starting estimate of the deformation field $u_0 = \mathbf{0}$,
     the image $I_2$ warped by this deformation field $I_2^{\text{warped}} = I_2$ and the
     array of filter sizes $r$ (see 4.1).
4  for $i$ in Number of filter sizes $r$ do
5      Set Sparse LAP parameters: $R = \mathrm{r}[i]$, $W = R$ and the subset $P_i$,
         not containing points within $W$ of borders.
6      for $j$ in Max. repeats do
7          Sparse Displacement Estimation: With the current $N$, $R$, $W$
             and $P_i$, calculate the deformation vectors $\Delta\mathrm{u}$ between $I_1$ and
             $I_2^{\text{warped}}$ using the Algorithm 4.
8          Global Fitting: Add the previous deformation estimate $u_{i-1}$ at
             $P_i$ to $\Delta\mathrm{u}$, and fit the result into a global deformation model,
             obtaining $\Delta u$ (see 4.3).
9          Update the Displacement Estimation: Set $u_i = u_{i-1} + \Delta u$.
10         Warp: Warp $I_2$ with $u_i$ to obtain a new $I_2^{\text{warped}}$.
11         [Optional] if $\mathrm{PSNR}\left(I_1, I_{2,j}^{\text{warped}}\right) - \mathrm{PSNR}\left(I_1, I_{2,j-1}^{\text{warped}}\right) > \epsilon$
             then
12             Break inner loop.
13         end
14     end
15 end
   Result: Deformation field aligning $I_2$ to $I_1$ and the registered source
           image, $I_2^{\text{warped}}$.
```

Remark 4.1. The default filter size pyramid r, is set in a similar manner as in the PF-LAP paper [5]. The largest filter half size R is set to $\frac{1}{4}$ of the smaller dimension of the source image, and then descends in powers of two until $R = 1$.

4.5 Speed Advantages

As discussed before in Section 4.2, besides the fact that the displacement is estimated only at a small fraction of points, which means that there are fewer systems of equations to prepare and solve, and fewer deformations to be extracted, there are no other implementation changes that proved to be useful. This however, is by itself enough to boost the speed of Sparse LAP considerably. This improvement can be seen in the the Listings 4.1 and 4.2, in the rows "prepare A and b", "solve linear systems" and "calculate flow".

The main source of speedup over the PF-LAP algorithm, is thanks to the fact that, with global fitting, there is no need for post-processing — no inpainting nor smoothing, which as seen in Listing 4.2 are responsible for great portions of the total runtime. Another significant factor slowing down PF-LAP, is pre-filtering, which, as demonstrated in the Chapter 5, doesn't need, to be effective, as it outperformed PF-LAP with the prefiltering option enabled.

The most time consuming process in both algorithms is filtering. In Sparse PF-LAP it represented 80% of the total time, while in PF-LAP 40%.

Remark. The input parameters for these runs were as follows; $N = 3$, Histogram matching = true, $Max.repeats = 3$ for both algorithms, specifically in Sparse PF-LAP, $D = 600$, $C = 13$ and quadratic global model fitting was used and specifically in PF-LAP, the prefiltering option was enabled.

Remark. The number of calls for PF-LAP was slightly higher, but this wasn't due to any parameter change, the PSNR testing process evaluated more iterations of PF-LAP as ones to be repeated.

Listing 4.1: Sparse PF-LAP timings

```
------------------------------------------------------------
                                                Time
                                      ----------------------
           Tot / % measured:               4.34s / 100%

Section                        ncalls     time   %tot     avg
------------------------------------------------------------
sparse pflap with psnr              1    4.33s   100%   4.33s
  single filter pyramid level       8    4.16s  95.9%   519ms
    sparse lap                     12    3.85s  88.7%   320ms
      filtering                    12    3.47s  80.2%   289ms
      prepare A and b              12    370ms  8.54%  30.8ms
        window sum part 1          36    241ms  5.57%  6.70ms
        window sum part 2          24    128ms  2.96%  5.35ms
      solve linear systems         12    601μs  0.01%  50.0μs
      calculate flow               12    121μs  0.00%  10.1μs
    image interpolation            12    158ms  3.66%  13.2ms
    flow interpolation             12    137ms  3.16%  11.4ms
  setup                             1    177ms  4.10%   177ms
    find edge points                1    109ms  2.51%   109ms
    hist match                      1   49.7ms  1.15%  49.7ms
------------------------------------------------------------
```

Listing 4.2: PF-LAP timings.

```
------------------------------------------------------------
                                                Time
                                       ----------------------
          Tot / % measured:                22.6s / 100%

Section                       ncalls     time   %tot      avg
------------------------------------------------------------
pflap                              1    22.6s   100%    22.6s
  single filter pyramid level      8    22.6s   100%    2.82s
    lap                           19    9.07s  40.1%    477ms
      filtering                   19    6.39s  28.3%    336ms
      prepare A and b             19    1.05s  4.63%   55.1ms
        window sum part 1         57    664ms  2.94%   11.7ms
        window sum part 2         38    227ms  1.00%   5.96ms
      solve linear systems        19   1000ms  4.42%   52.6ms
      calculate flow              19    134ms  0.59%   7.05ms
    prefiltering                  46    6.78s  30.0%    147ms
    smoothing                     19    3.95s  17.5%    208ms
    inpainting                    19    2.34s  10.3%    123ms
      replicating borders         19   54.7ms  0.24%   2.88ms
    image interpolation           19    340ms  1.50%   17.9ms
  setup                            1   46.7ms  0.21%   46.7ms
    hist match                     1   40.9ms  0.18%   40.9ms
------------------------------------------------------------
```

Chapter 5

Experiments

In this chapter, I evaluate the performance of the proposed algorithm, Sparse PF-LAP, against the following image registration algorithms:

1. **BUnwarpJ** [17, 18]: an ImageJ/Fiji image registration plugin, which estimates a B-spline-based deformation. It uses multi-resolution and the minimised criterion is a sum of squares difference (SSD).

2. **Elastix** [19, 20]: an image registration software base on the Insight Segmentation and Registration Toolkit (ITK). It consists of a collection of algorithms, in the following experiments b-spline image registration in a multi-resolution scheme was used.

3. **DROP2** [21, 22]: an intensity-based image registration toolkit with both linear and non-linear registration methods. In the experiments both linear and non-linear registration was used.

4. **PF-LAP** [5]: an intensity-based image registration algorithm with a multi-resolution scheme. Discussed in Chapter 3.

Remark. There are more registration algorithms provided in BIRL, but their results on the chosen testing datasets were not good, so I excluded them from the experiments.

To run these registration methods, I use the image registration benchmark framework provided from the Automatic Non-rigid Histological Image Registration (ANHIR) Challenge [23], the Benchmark on Image Registration methods with Landmark validation (BIRL) [24, 25]. The ANHIR challenge, and its' benchmarking framework BIRL, use Euclidean distance of two corresponding landmarks Target Registration Error (TRE) as a performance metric, more on this in the following paragraph.

BIRL's performance metrics. Every corresponding image pair I_1, I_2 in the histology dataset used in ANHIR has a number of corresponding key-points that, when the images I_1 and I_2 get geometrically aligned, should overlap. In other words, when a registration algorithm aligns I_2 with I_1, it also aligns key-points of I_2 with the ones in I_1. BIRL uses the misalignment of the key-points as a performance metric and terms it Target Registration

Error (TRE), defined as the Euclidean distance between the key-point of the registered source image I_2^{reg} and the key-point in target image I_1. A mean or median can then be taken from the TRE of all key-points of an image. In the following tests, I use mean and median TRE as a performance metric for the registration of one image, more on this in the following Section 5.1.

CIMA histology dataset. First, I tested the PF-LAP algorithm and the proposed methods on the histology dataset from the Center for Applied Medical Research (CIMA)[25, 26, 27] containing 2D histological microscopy tissue slices, stained with different stains and with annotated landmarks at key-points in each slice, as used in the ANHIR challenge. But the results of the PF-LAP and the Sparse PF-LAP algorithms were not good, revealing a shortcoming of this method. One reason was probably that the images in this dataset were not ideal for the LAP approach. Even though the images in this dataset are technically mono-modal, they can be assumed to be almost multi-modal due the fact that the variety of the used stains dramatically change the appearance model, and the deformation may range from fine elastic transformation to completely missing sections. For this reason I tested the methods on two other datasets, consisting of real mono-modal images.

1. One experiment is performed on the Oxford affine dataset [28], which has images covering a range of situations like blurring, varying illumination and change in viewpoint. These images are provided with a ground truth homography matrix, see Section 5.3.

2. And another experiment is performed on a subset of images taken from the dataset of the AAPM RT-MAC Grand Challenge 2019 [14, 15], which consists of head and neck MR scans which I artificially warped with random homography transformations, see Section 5.4.

Remark. All tests were run on the docker image provided in BIRL on a computer with Intel Core i7 at 3,4 GHz with 16GB of RAM.

5.1 Performance Metrics

When the ground truth deformation is known (as in the two experiments described above), an ideal performance metric of an image registration algorithm could be a mean or median absolute deformation error. Comparing image registration algorithms from different toolkits or as standalone programs in different programming languages, can be hard however, because the representation of the outputted deformation can be unique to each of the methods. For this reason, I chose to use BIRL as medium to compare the proposed methods to others. But since BIRL does not measure error of the dense displacement field, and the datasets above do not have annotated key-points, I create a large number of uniformly distributed key-points in I_1 and shift them with the known ground truth deformation field to create corresponding key-points in I_2. This gives me the ability to measure the TRE of key-points of the registered source image I_2^{reg} and the target image I_1.

I chose to use mean TRE, TRE_{Mean} and median TRE, TRE_{Med} and execution time (in seconds) as the three quality measurements to compare the algorithms on one image pair. When assessing performance on the datasets, I take the mean of these measurement, effectively measuring average TRE_{Mean}, average TRE_{Med} and average time.

Remark. Note that the more key-points used, the closer the TRE is to the absolute deformation error.

5.2 Method Parameters

All of the tested methods have configurable input parameters to best suit the type of registration. For tests on both datasets, Oxford affine and head MR images, I set the transformation parameter to affine for Elastix, DROP2 was set to use both linear and non-linear registration and the parameters of BUnwarpJ and other parameters of the first two methods were left as defaults from the BIRL framework. As for the LAP based methods, the number of base filters N was set to 3, and histogram matching was used as a part of the procedure. PF-LAP was run both with, and without prefiltering. For the Sparse methods, the number of edge points D was set to 500, the exclusion radius C to 13 and quadratic model fitting was used, as it better suits the ground truth deformation.

5.3 Experiment on the Oxford Affine Dataset

In this experiment I show how the proposed algorithm copes, when brightness constancy is violated. I chose four subsets of images from the Oxford affine dataset [28]:

1. **Bikes**, containing images of size 1000×700 pixels, in which the source images are corrupted by blurring and have a maximum displacement ranging from 39.2 to 52.7 pixels,

2. **Leuven**, containing images of size 900×600 pixels, in which the source images are corrupted by illumination change and have a maximum displacement ranging from 7.7 to 22.2 pixels,

3. **Trees**, containing images of size 1000×700 pixels, in which the source images are corrupted by blurring and by the movement of leaves and have a maximum displacement ranging from 38.6 to 55.4 pixels,

4. **UBC**, containing images of size 800×640 pixels, in which the source images are corrupted by JPEG compression and do not have a displacement at all.

Each subset contains a one target image I_1 and five source images I_2, I_3, I_4, I_5 and I_6 — from small changes in the imaging conditions to large changes. The dataset contains a ground truth homography transformation from the target image to all of the source images for each subset. In section A.1 of the Appendix A, there is a figure for each subset showing I_1, I_3, the ground truth deformation and the deformation estimated by the Sparse PF-LAP method.

Remark. There are more subsets in this dataset, but the maximum displacement is too high for the PF-LAP, Sparse PF-LAP and the other tested methods, to output a meaningful registration.

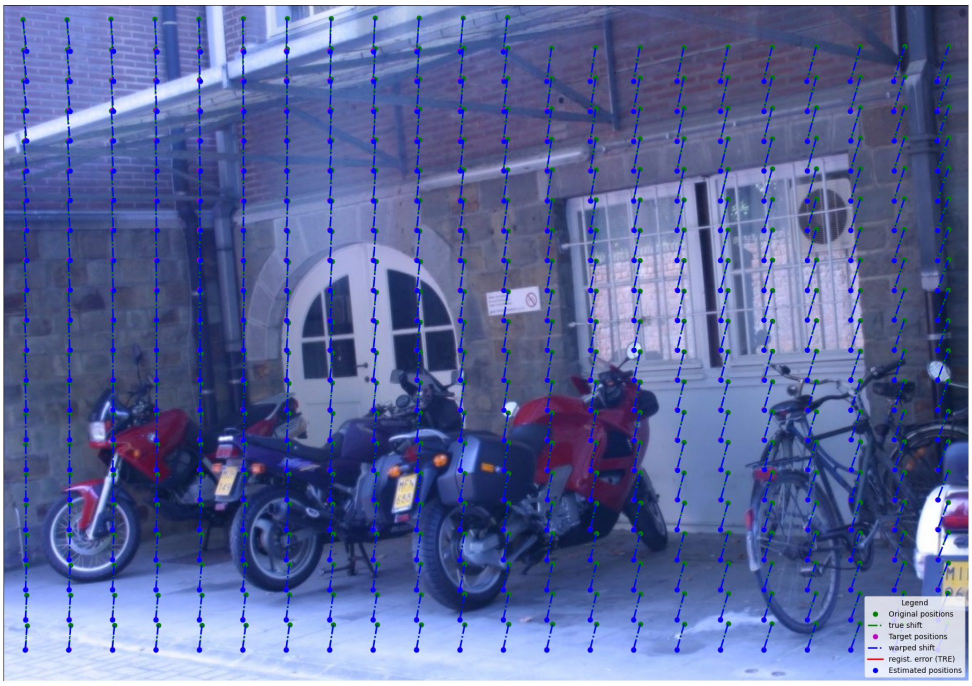


Figure 5.1: An well registered source image from the Bikes subset of the Oxford affine dataset, with estimated and target key-points.

	Method	Avg. TRE$_{\text{Mean}}$	Avg. TRE$_{\text{Med}}$	Avg. Time
Bikes	DROP2	6.83	6.68	0.981
	Elastix	**0.811**	**0.699**	63.7
	BUnwarpJ	2.59	2.49	66.4
	Sparse PF-LAP	0.999	0.944	9.07
	Sparse PF-LAP$_{\text{PSNR}}$	1.15	0.993	14.7
	PF-LAP	6.21	3.68	29.3
	PF-LAP$_{\text{prefilt}}$	1.95	1.65	4.42
	Sparse PF-LAP*	1.36	1.17	**0.964**
	Sparse PF-LAP$_{\text{PSNR}}{}^{*}$	1.11	0.995	1.56
	PF-LAP*	6.99	4.03	1.99
	PF-LAP$_{\text{prefilt}}{}^{*}$	2.0	1.64	2.1
Leuven	DROP2	1.87	1.9	**0.792**
	Elastix	0.797	0.526	63.5
	BUnwarpJ	2.41	2.37	72.6
	Sparse PF-LAP	**0.38**	**0.261**	8.03
	Sparse PF-LAP$_{\text{PSNR}}$	0.547	0.413	11.7
	PF-LAP	1.5	0.805	18.5
	PF-LAP$_{\text{prefilt}}$	1.15	0.842	3.35
	Sparse PF-LAP*	0.419	0.371	0.955
	Sparse PF-LAP$_{\text{PSNR}}{}^{*}$	0.496	0.396	1.01
	PF-LAP*	2.41	1.51	1.34
	PF-LAP$_{\text{prefilt}}{}^{*}$	1.53	1.14	1.68
Trees	DROP2	5.49	5.51	1.22
	Elastix	**1.69**	**1.3**	64.1
	BUnwarpJ	3.98	3.26	67.5
	Sparse PF-LAP	3.66	2.62	9.37
	Sparse PF-LAP$_{\text{PSNR}}$	2.54	2.12	15.7
	PF-LAP	6.1	3.11	29.5
	PF-LAP$_{\text{prefilt}}$	4.97	2.73	4.22
	Sparse PF-LAP*	2.94	2.41	**0.974**
	Sparse PF-LAP$_{\text{PSNR}}{}^{*}$	2.51	2.09	1.4
	PF-LAP*	5.67	3.41	1.59
	PF-LAP$_{\text{prefilt}}{}^{*}$	5.05	2.86	2.12
UBC	DROP2	**0.0636**	0.0635	**0.763**
	Elastix	0.122	0.114	65.0
	BUnwarpJ	2.97	2.63	71.5
	Sparse PF-LAP	0.0811	**0.0612**	6.91
	Sparse PF-LAP$_{\text{PSNR}}$	0.157	0.123	10.7
	PF-LAP	8.15	2.01	29.3
	PF-LAP$_{\text{prefilt}}$	2.19	0.897	3.36
	Sparse PF-LAP*	1.21	0.521	1.25
	Sparse PF-LAP$_{\text{PSNR}}{}^{*}$	0.566	0.417	1.14
	PF-LAP*	7.57	1.79	2.27
	PF-LAP$_{\text{prefilt}}{}^{*}$	1.9	0.873	1.47

Table 5.1: Results on the Oxford affine dataset. Best values in each subset are highlighted in bold. Methods marked with (*) are run on images resized to 400 diagonal pixels (see section 5.3).

Down-scaling. Since the DROP2, Elastix and BUnwarpJ methods all use image down-scaling in the form of a multi-resolution scheme, I decided to also downscale the images to show how the proposed methods cope with this. The images were down-scaled to from roughly 1100 diagonal pixels, depending on the subset, to 400 diagonal pixels. These runs that used down-scaled images, are marked the (*) in the results Table 5.1.

Results

The results of the compared methods are displayed in Table 5.1. Best values are highlighted in bold. The Sparse PF-LAP method performed best in the Leuven and UBC subsets, where the displacement is the smallest, achieving sub-pixel precision. In the Leuven dataset it was closely followed by the down-scaled run of the algorithm, while in the UBC the first place in precision was split between it and DROP2, which was quite a bit faster.

On original sized images in the Leuven, Bikes and UBC subsets, Sparse PF-LAP without PSNR testing came out more precise (almost twice as precise in UBC), than Sparse PF-LAP with PSNR testing, but performed worse in the Trees subset.

Interestingly, but for the Leuven dataset — where it was otherwise — the Sparse PF-LAP with PSNR testing outperformed the Sparse PF-LAP method on all of the down-scaled images — in the UBC and Trees subset the difference was significant.

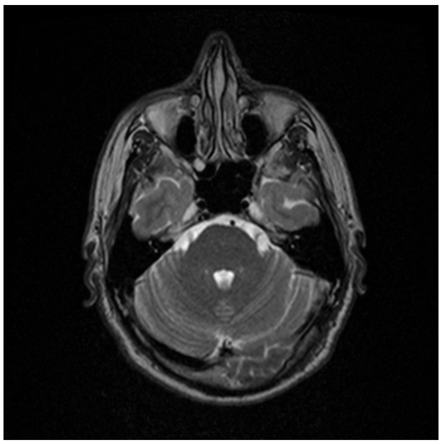
The timings of all methods were quite stable—varying by a small amount over all four subsets. Tied for fastest is the down-scaled Sparse PF-LAP and DROP2 at roughly 1.1 seconds of runtime, with the down-scaled Sparse PF-LAP with PSNR testing close behind. While the slowest by far are Elastix and BUnwarJ with around 65 seconds of execution time.

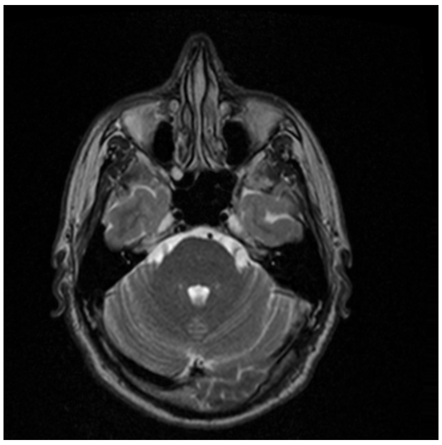
On the whole, both of the proposed algorithms outperformed PF-LAP (with, and without prefiltering) in all the performance metrics measured. The methods came out in second place in the Bikes subset, where they were slightly outperfored by Elastix. And they managed second place in the Trees subset where the Elastix algorithm dominated every other, beating Sparse PF-LAP by a large margin.

5.4 Experiment on Head MR Images

In this experiment the algorithms are run on 60 MR scans of the head and neck, from the AAPM RT-MAC Grand Challenge 2019 dataset [14, 15]. These target images were filled with key-points and deformed with a random homography transform to create the source images. Each transform was rescaled, so that the maximum displacement is 40 pixels. An example of a target and source image is in Figure 5.2.

Remark. The brightness constancy not violated in this experiment.


(a) : Target image I_1.


(b) : Source image I_2.

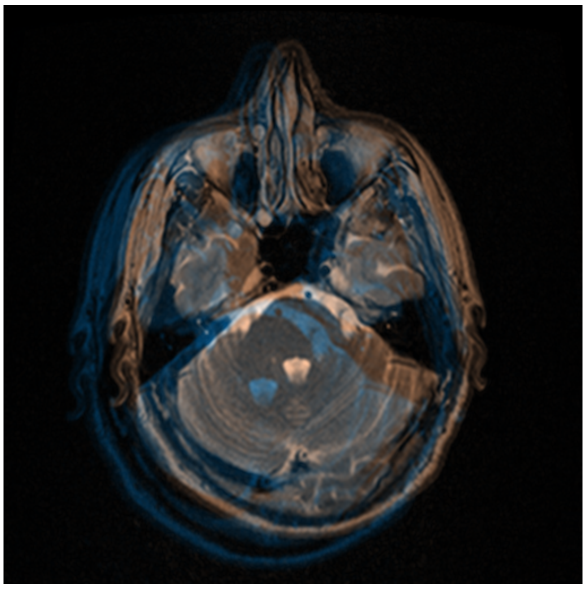

(c) : I_1 (orange) and I_2 (blue).

Figure 5.2: MR head scan: target and generated source.

Down-scaling. Similarly to experiment on the previous dataset, I use run the LAP based methods on down-scaled images. Here the down-scaling factor was smaller in comparison; from 724 diagonal pixels to 400.

Results

The results of the compared methods are displayed in Table 5.2.

In this experiment Elastix is the clear winner quality-wise, outperforming all other algorithms by a good margin, but speed-wise it is lacking, being the slowest after BUnwarpJ.

Since, discounting noise and scan artifacts, the margins where the image is only black are quite wide, I thought this would better suite the Sparse algorithms over PF-LAP. But surprisingly, PF-LAP performed quite well on these images. With prefiltering it achieved an average $\mathrm{TRE}_{\mathrm{Med}}$ of 2.04, Sparse PF-LAP followed with 2.23. In average TRE_{Mean} however, Sparse PF-LAP came in second with PF-LAP third.

The runtimes on this data are similar as in the previous one, the down-scaled Sparse PF-LAP being the fastest with 1.23 seconds and followed by the down-scaled Sparse PF-LAP with PSNR testing with 1.77 seconds.

Unfortunately, DROP2 is excluded from the results, because every registration attempt ended with an internal error.

Method	Avg. TRE_{Mean}	Avg. TRE_{Med}	Avg. Time
DROP	13.9	13.3	**0.491**
Elastix	**1.36**	**1.16**	70.5
BUnwarpJ	270.0	185.0	74.6
Sparse PF-LAP	2.54	2.23	3.9
Sparse $\text{PF-LAP}_{\text{PSNR}}$	2.74	2.34	6.38
PF-LAP	11.2	3.93	17.9
$\text{PF-LAP}_{\text{prefilt}}$	2.79	2.04	5.85
Sparse PF-LAP*	3.48	2.89	1.23
Sparse $\text{PF-LAP}_{\text{PSNR}}{}^*$	2.67	2.32	1.77
PF-LAP*	11.4	4.23	3.34
$\text{PF-LAP}_{\text{prefilt}}{}^*$	2.88	2.07	3.92

Table 5.2: Results on head MR images. Best values are highlighted in bold. Methods marked with (*) are run on images resized to 400 diagonal pixels (see section 5.4).

Chapter 6

Summary and Conclusion

In this thesis, I first briefly explained what image registration is, and what are some approaches to solve the image registration problems. Then I described in detail how the Poly-Filter Local All-Pass (PF-LAP) and Local All-Pass (LAP) image registration algorithms work. After that, I proposed a new method—Sparse Poly-Filter Local All-Pass (Sparse PF-LAP)—based the PF-LAP algorithm, that uses sparse displacement estimation and global deformation model fitting. And compared this methods experimentally with others, on real, and artificial data.

The proposed method performed very well in both of the conducted experiments, in quality, outperforming PF-LAP and all other tested methods, besides Elastix, which was superior in on some data. Sparse PF-LAP was also shown to be very fast. It was faster than PF-LAP by a good margin, and, with image down-scaling, was faster than every other algorithm, except for DROP2. The performance of the proposed methods was very good even on down-scaled images, even improving the registration quality on the Trees subset of the Oxford affine dataset. On the whole, the sparse estimation and global fitting approach was shown to be very effective, and I think it should be explored further. In particular, a multi-resolution scheme could be implemented, to improve the speed for large images, while keeping the registration quality high. Another area left unexplored, is to improve the algorithm to work on multi-modal-like images, similar to the ones from the ANHIR challenge.

Appendix A
Datasets

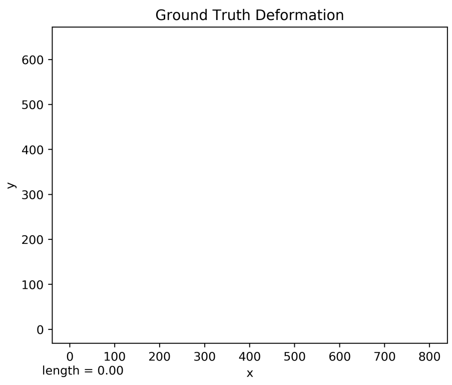
Examples of images from the datasets used in Chapter 5.

A.1 Oxford Affine

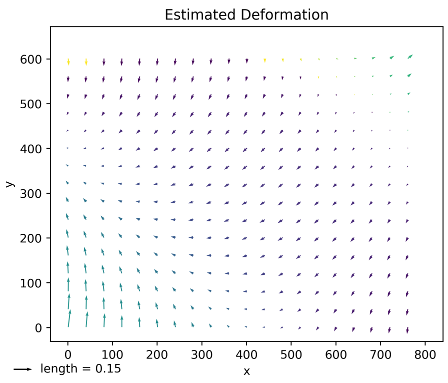
Examples of dataset images used in the first experiment in Chapter 5, target and source from each subset, along with the ground truth deformation and deformation estimated by the Sparse PF-LAP algorithm without PSNR testing.

(a) : I_1

(b) : I_3



(c) : u

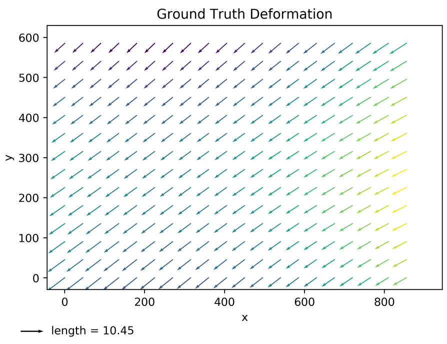


(d) : u_{est}

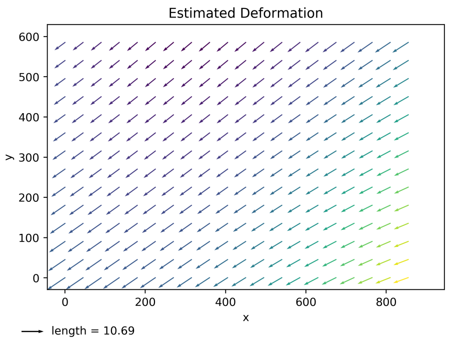
Figure A.1: UBC subset.

(a) : I_1

(b) : I_3

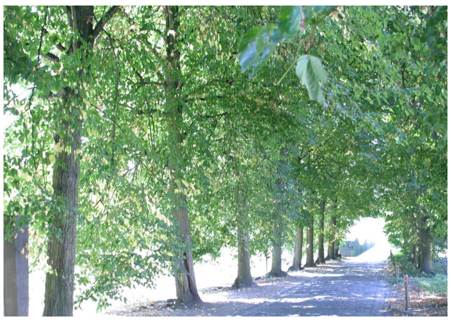


(c) : u

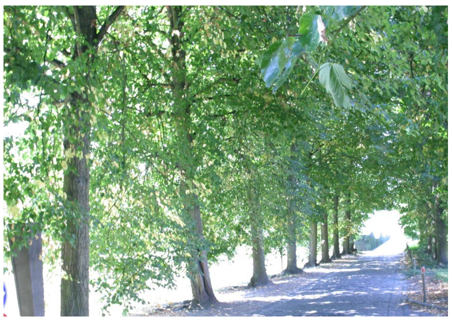


(d) : u_{est}

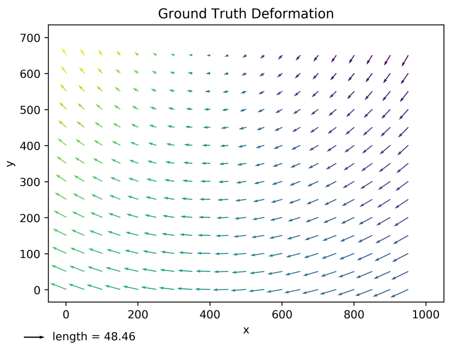
Figure A.2: Leuven subset.



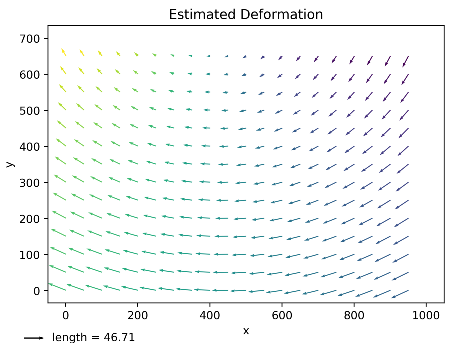
(a) : I_1



(b) : I_3



(c) : u

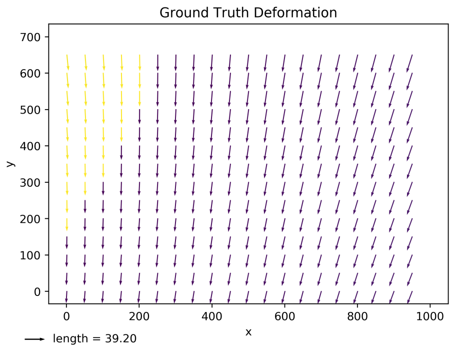


(d) : u_{est}

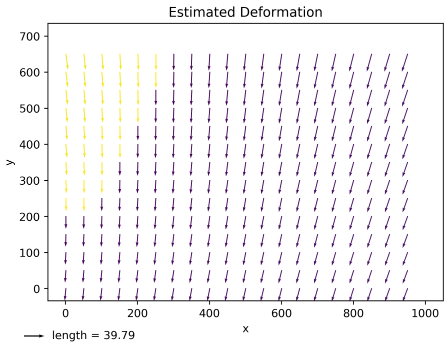
Figure A.3: Trees subset.

(a) : I_1

(b) : I_3



(c) : u

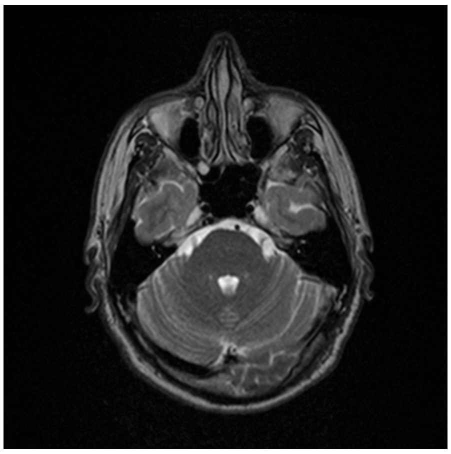


(d) : u_{est}

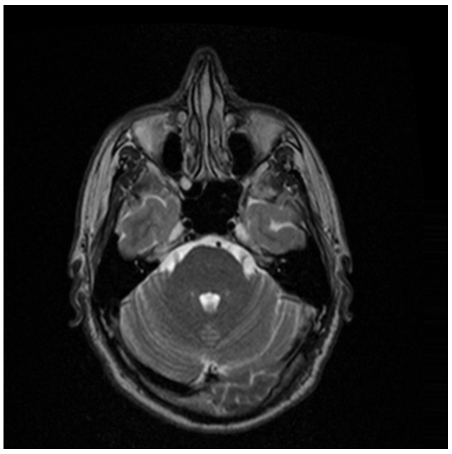
Figure A.4: Bikes subset.

A.2 MR head images

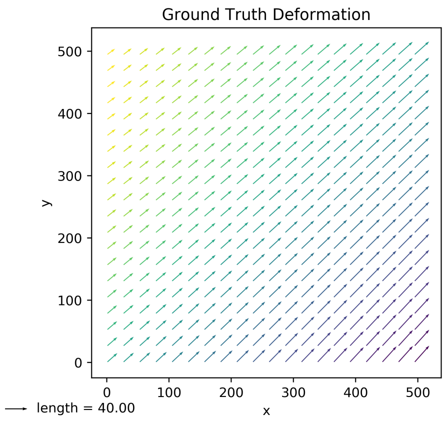
One random example of dataset images used in the second experiment in Chapter 5, along with the ground truth deformation and deformation estimated by the Sparse PF-LAP algorithm without PSNR testing.



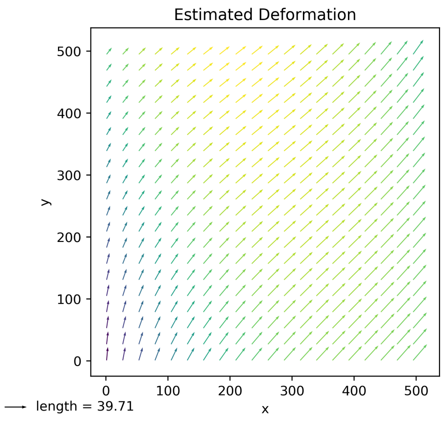
(a) : I_1



(b) : I_2



(c) : u



(d) : u_{est}

Figure A.5: MR head images.

Appendix B
Attachment contents

```
/
├── BIRL ....................... used datasets, run scripts and test results
├── thesis.pdf .................... pdf file with the thesis text
├── LAP_julia ................. Julia package with implemented methods
└── README.md .................. text file with the GitHub and other urls
```

Appendix C

Bibliography

[1] Josien Pluim, J. Maintz, and Max Viergever. Mutual-information-based registration of medical images: A survey. *Medical Imaging, IEEE Transactions on*, 22:986 – 1004, 09 2003.

[2] X. Zhang, C. Gilliam, and T. Blu. Parametric registration for mobile phone images. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1312–1316, 2019.

[3] Barbara Zitová and Jan Flusser. Image registration methods: A survey. *Image and Vision Computing*, 21:977–1000, 10 2003.

[4] Jan Kybic and Jiří Borovec. Fast registration by boundary sampling and linear programming. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pages 783–791. Springer International Publishing, 2018.

[5] Christopher Gilliam and Thierry Blu. Local all-pass geometric deformations. *IEEE Transactions on Image Processing*, PP:1–1, 10 2017.

[6] Christopher Gilliam and Thierry Blu. Local all-pass filters for optical flow estimation. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, April 2015.

[7] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, Mar 2011.

[8] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[9] Xinxin Zhang, Christopher Gilliam, and Thierry Blu. All-pass parametric image registration. *IEEE Transactions on Image Processing*, PP, 04 2020.

[10] Jan Kybic. Registration of segmented histological images using thin plate splines and belief propagation. In Sebastien Ourselin and Martin A.

Styner, editors, *Medical Imaging 2014: Image Processing*. SPIE, March 2014.

[11] Manuel M. Oliveira, Brian Bowen, Richard McKenna, and Yu sung Chang. Fast digital image inpainting. In *PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON VISUALIZATION, IMAGING AND IMAGE PROCESSING (VIIP 2001*, pages 261–266. ACTA Press, 2001.

[12] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA, 2006.

[13] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988.

[14] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The cancer imaging archive (tcia): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, Dec 2013.

[15] Carlos Cardenas, Abdallah Mohamed, Greg Sharp, Mark Gooding, Harini Veeraraghavan, and Yang Jinzhong. Data from aapm rt-mac grand challenge 2019, 2019.

[16] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive Theory of Functions of Several Variables*, pages 85–100. Springer Berlin Heidelberg, 1977.

[17] C.Ó. Sánchez Sorzano, P. Thévenaz, and M. Unser. Elastic registration of biological images using vector-spline regularization. *IEEE Transactions on Biomedical Engineering*, 52(4):652–663, April 2005.

[18] Ignacio Arganda-Carreras, Carlos O. S. Sorzano, Roberto Marabini, José María Carazo, Carlos Ortiz de Solorzano, and Jan Kybic. Consistent and elastic registration of histological sections using vector-spline regularization. In *Computer Vision Approaches to Medical Image Analysis*, pages 85–95. Springer Berlin Heidelberg, 2006.

[19] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. W. Pluim. elastix: A toolbox for intensity-based medical image registration. *IEEE Transactions on Medical Imaging*, 29(1):196–205, 2010.

[20] Denis Shamonin, Esther Bron, Boudewijn Lelieveldt, Marion Smits, Stefan Klein, and Marius Staring. Fast parallel image registration on cpu and gpu for diagnostic classification of alzheimer's disease. *Frontiers in neuroinformatics*, 7:50, 01 2013.

[21] Ben Glocker, Nikos Komodakis, Georgios Tziritas, Nassir Navab, and Nikos Paragios. Dense image registration through mrfs and efficient linear programming. *Medical image analysis*, 12(6):731–741, 2008.

[22] Ben Glocker, Aristeidis Sotiras, Nikos Komodakis, and Nikos Paragios. Deformable medical image registration: setting the state of the art with discrete methods. *Annual review of biomedical engineering*, 13:219–244, 2011.

[23] 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. Yushkevic, S. Zhao, and A. Muñoz-Barrutia. Anhir: Automatic non-rigid histological image registration challenge. *IEEE Transactions on Medical Imaging*, pages 1–1, 2020.

[24] Jiri Borovec. Birl: Benchmark on image registration methods with landmark validation. *ArXiv*, abs/1912.13452, 2019.

[25] Jiri Borovec, Arrate Munoz-Barrutia, and Jan Kybic. Benchmarking of Image Registration Methods for Differently Stained Histological Slides. In *IEEE International Conference on Image Processing (ICIP)*, pages 3368–3372, Athens, 2018.

[26] Rodrigo Fernandez-Gonzalez, Arthur Jones, Enrique Garcia-Rodriguez, Ping Chen, Adam Idica, Stephen Lockett, Mary Barcellos-Hoff, and Carlos Ortiz-de Solorzano. System for combined three-dimensional morphological and molecular analysis of thick tissue specimens. *Microscopy research and technique*, 59:522–30, 12 2002.

[27] Jiri Borovec, Jan Kybic, Michal Busta, Carlos Ortiz de Solorzano, and Arrate Munoz-Barrutia. Registration of multiple stained histological sections. In *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE, April 2013.

[28] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1):43–72, Nov 2005.