UNIVERSITÀ DEGLI STUDI DI PALERMO

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

DIPARTIMENTO DI INGEGNERIA INFORMATICA

FUZZY KERNEL REGRESSION FOR REGISTRATION AND OTHER IMAGE WARPING APPLICATIONS

Tutor
Prof. Roberto Pirrone

Candidato
Ing. Roberto Gallea

Coordinatore
Ch.mo Prof. Salvatore Gaglio

Tesi di Dottorato di Ricerca in Ingegneria Informatica – XXII Ciclo
SSD: ING-INF/05

Fuzzy Kernel Regression for Registration and other Image Warping Applications

Roberto Gallea

Dipartimento di Ingegneria Informatica

Universita' degli Studi di Palermo

A thesis submitted for the degree of

Doctor of Philosophy

15/02/2011

To my bride-to-be, Renata.

To my loving parents.

To me (too).

Abstract

Supervising Professor: *Roberto Pirrone*

In this dissertation a new approach for non-rigid medical image registration is presented. It relies onto a probabilistic framework based on the novel concept of Fuzzy Kernel Regression. The theoric framework, after a formal introduction is applied to develop several complete registration systems, two of them are interactive and one is fully automatic. They all use the composition of local deformations to achieve the final alignment. Automatic one is based onto the maximization of mutual information to produce local affine aligments which are merged into the global transformation. Mutual Information maximization procedure uses gradient descent method. Due to the huge amount of data associated to medical images, a multi-resolution topology is embodied, reducing processing time. The distance based interpolation scheme injected facilitates the similairity measure optimization by attenuating the presence of local maxima in the functional. System blocks are implemented on GPGPUs allowing efficient parallel computation of large 3d datasets using SIMT execution. Due to the flexibility of Mutual Information, it can be applied to multimodality image scans (MRI, CT, PET, etc.).

Both quantitative and qualitative experiments show promising results and great potential for future extension.

Finally the framework flexibility is shown by means of its succesful application to the image retargeting issue, methods and results are presented.

Acknowledgements

I would like to acknowledge Dr. Filippo Barbiera, director of the Radiology Department of "Azienda Ospedaliera Ospedali Riuniti", Sciacca (AG), Italy, and his staff for providing their expertise and support for results evaluation and their valuable suggestions.

I would like also to thank Nvidia for providing support to the project by donating a Tesla C2070 GPU card, allowing me to make development, tests and experiments.

Finally, my last acknowledgements go to Prof. Roberto Pirrone and Prof. Edoardo Ardizzone, who supported me in my research during these three years. And to Marco, Massimo, Tonino and Orazio, who shared the same path with me, both difficulties and successes, thank you.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AAB Automatic area based registration system

AID Average Intensity Difference error measure

DI Distance Interpolation

DIJIT Jittered Distance Interpolation

EMD Earth Mover's Distance

FCM Fuzzy c-means clustering

GPGPU General Purposes GPUs

GPU Graphics Processing Unit devices

I Input image (also called base or moving)

ILB Improved landmark based registration system

LI Linear Interpolation

MI Mutual Information

NMI Normalized Mutual Information

NN Nearest Neighbor interpolation

NNJIT Jittered Nearest Neighbor Interpolation

PVI Partial Volume Interpolation

RMS Root Mean Square error measure

SIFTflow SIFT (Scale-invariant feature transform) flow distance

SIMT Single Instruction Multiple Thread execution scheme, used in CUDA GPUs

SLB Simple landmark based registration system

T Target image (also called reference or fixed)

Chapter 1

Introduction

In the current clinical setting, medical imaging is an umprescindible component in a large number of applications throughout the whole track of events. It is not required just for diagnostic purposes, but it is equally prominent in the planning, execution and evaluation of surgical and therapeutical procedures. There exist two categories of medical imaging: *anatomical* and *functional*. Anatomical, exhibiting mainly morphological structures, include X-ray, CT (computed tomography), MRI (magnetic resonance imaging), US (ultrasound) and others such as laparoscopy and laryngoscopy. Other can be derived from the previous, such as MRA (magnetic resonance angiography), DSA (digital subtraction angiography derived from X-ray), CTA (computed tomography angiography), and *Doppler* (derived from US). From the other hand, functional imaging, provides information about the metabolism underlying anatomy, and include scintigaphy, SPECT (single photon emission computed tomography), PET (positron emission tomography) which belong to the *nuclear medicine* image modalities, and fMRI (functional MRI). Other additional modalities exist, but they are poorly used or are currently in a research state.

Since information gained from two different clinical tracks are usually complementary, proper integration of merged data can be desired. The first step of such integration, which is the spatial alignment of the datasets, is referred as *registration*.

In addition to different modalities, the datasets to be aligned can be provided by sequential acquisitions too. This can be useful to determine the variations of

some structures during time, for example due to the effects of a therapy, or pre- and post-intervention imaging.
Image registration aims to obtain the best possible spatial correspondence between misaligned datasets, namely the floating (or moving) image, and the reference (or fixed) image. This procedure is also useful to correct distortions induced by magnetic interferences with the acquisition equipment signals or the ones due to patient's involuntary movements such as heartbeat or breathing.

1.1 Motivation and goals

Listing the benefits of systems related to diagnosis support is a trivial task, since the importance of precise and accurate diagnosis is obvious. Among these, medical image registrations systems are versatile and can serve to several goals.

- Gain additional information from the data-fusion on multi-modal scans of the same patient. This allows more accurate diagnosis and internvent planning for treatment by the clinician in order to obtain a detailed representation of patient's anatomy.
- Observe the changes in the patient's scans after a time period, due to a therapy or surgical operation.
- Map the anatomical structure of the patient to an atlas in order to evaluate more easily the morphology.
- Compare different heterogeneous datasets to accomplish several operations, such as Content Based Image Retrieval.

The goal of the project is to design a generic framework and implement systems for multi-modal brain images registration. Care is devoted to efficiency using optimizations and parallel computer architectures.

1.2 Fuzzy Kernel Regression for Registration

The approach followed in this work for elastic image registration is based on the hierarchic composition of local transformations. Regions of the floating image are locally aligned to the reference image and the global transformation is recovered blending the registered subimages. The method used for such composition is the "Fuzzy Kernel Regression", a novel technique for regression based onto two well-established pattern recognition concepts: fuzzy c-means clustering and kernel regression. Dense c-means memebership values (called *membership maps*) are used as equivalent kernels for the regression procedure. In this way kernel shapes are computed adaptively to the known data points without any optimization. The whole framework is set up into a probabilistic context. Due to its flexibility it represents a general basis for the construction of registration systems of different nature.

1.3 Contributions

The framework, formally stated, can be used for image registration and other regression- or interpolation-based purposes as well. It was applied to design and develop several registration systems, which have been tested both quantitatively and qualitatively to assess their effectiveness and performance. In addition, due to the extremely intensive data-based nature of such procedures, critical implementation issues were faced using GPGPU-based parallel solutions. The main work result is a general reusable framework and a set of systems. In addition, the building blocks used, such as fuzzy kernel regression, GPU-based fuzzy c-means and interpolation scheme, were extensively tested and individually evaluated. As a result, they can be plugged into other systems to improve their performance. This versatility is demonstrated with the application of such concepts to other image processing problems, and the succesful case of content-aware image resizing (or *retargeting*) is presented.
Resuming, the work done consists in the following:

- **Fuzzy Kernel Regression (FKR) framework statement**: the theoric basis of the concepts involved in this dissertation.

- **Three image registration systems**:
  - *Simple landmark based (SLB)*: simple and straightforward registration system deriving from FKR formulation.
  - *Improved landmark based (ILB)*: improves the previous using a region-wise approach for registration.
  - *Automatic area based (AAB)*: fully automatic registration system based on Normalized Mutual Information maximization.
- **Additional optimization and improvements**: several tuning and optimizations are made to keep computation of very large datasets affordable and efficient.
  - *GPUs-based implementations*: Mutual Information estimation and Fuzzy c-means clustering are implemented using GPU devices.
  - *Further operators enhancements*: additional improvements are devoted to most used operators.
- **Application of FKR to retargeting**: FKR framework is tested on different image processing problems, the case of retargeting is reported.
  - *Retargeting system design and implementation*: formulation and realization of a retargeting system based on linear optimization and FKR.
  - *Web implementation*: implementation of the system as a web service transparent to the developer.

1.4 Publications

During the Ph.D. course, several works related to this research activity were published in the field of image processing and medical imaging:

- **Medical Image Registration: interpolations, similarities and optimizations strategies** - *R. Gallea, E. Ardizzone, R. Pirrone, O. Gambino* - CBMS 2010 - IEEE International Symposium on Computer-Based Medical Systems, Oct 12-15, 2010, pp. 72-77, ISBN 978-1-4244-9166-7

- **Effective and Efficient Interpolation for Mutual Information based Multimodality Elastic Image Registration** - *R. Gallea, E. Ardizzone, R. Pirrone, O. Gambino* - ICCV 2009 - 2009 IEEE International Conference on Computer Vision, Sept 27-Oct 4, 2009, pp. 376-381, ISBN 978-1-4244-4442-7, DOI 10.1109/ICCVW.2009.5457677

- **Multi-modal Image Registration using Fuzzy Kernel Regression** - *R. Gallea, E. Ardizzone, R. Pirrone, O. Gambino* - ICIP 2009 - 2009 IEEE International Conference on Image Processing, 7-10 Nov. 2009, pp. 193-196, ISBN 978-1-4244-5653-6, ISSN 1522-4880, DOI 10.1109/ICIP.2009.5414220

- **Fuzzy Smoothed Composition of Local Mapping Transformations for Non-rigid Image Registration** - *R. Gallea, E. Ardizzone, R. Pirrone, O. Gambino* - ICIAP 2009 - 15th International Conference on Image Analysis and Processing, LNCS 5716, Sep 8- 11 2009, pp. 777-786, ISSN 0302-9743, DOI: 10.1007/978-3-642-04146-4_83

- **Fuzzy C-Means Inspired Free Form Deformation Technique for Registration** - *R. Gallea, E. Ardizzone, R. Pirrone, O. Gambino* - WILF 2009 - International Workshop on Fuzzy Logic and Applications, LNCS 5571, Jun 9-12 2009, pp. 148-154, ISSN 0302-9743, DOI 10.1007/978-3-642-02282-1_19

- **A combined Fuzzy and Probabilistic data descriptor for distributed CBIR** - *R. Gallea, M. La Cascia, M. Morana* - WILF 2009 - International Workshop on Fuzzy Logic and Applications, LNCS 5571, Jun 9-12 2009, pp. 189-196, ISSN 0302-9743, DOI 10.1007/978-3-642-02282-1_24

- **Noise Filtering Using Edge-Driven Adaptive Anisotropic Diffusion** - *E. Ardizzone, R. Pirrone, R. Gallea and O. Gambino* - Proceedings of CBMS 2008, 21th IEEE International Symposium on Computer-Based Medical Systems, Jun 17-19, pp. 29-34, ISBN 978-0-7695-3165-6, ISSN 1063-7125, DOI 10.1109/CBMS.2008.31

- **Fuzzy-based kernel regression approaches for free form deformation and elastic registration of medical images** - *R. Gallea, E. Ardizzone, R. Pirrone, O. Gambino* - Chapter of Recent Advances in Biomedical Engineering - IN-TECH (2009), pp. 347-368, ISBN: 978-953-307-013-1

- **Browser independent content based image resizing for liquid web layouts** - *R. Gallea, E. Ardizzone, R. Pirrone* - SAPMIA 2010 - International Workshop on Social, Adaptive and Personalized Multimedia Interaction and Access, Oct 25-29, 2010, pp. 29-32, ISBN 978-1-60558-933-6

- **Real-time content-aware image resizing using reduced linear model** - *R. Gallea, E. Ardizzone, R. Pirrone* - ICIP 2010 - 2010 IEEE International Conference on Image Processing, Sep 26-29, 2010, pp. 2813-2816, ISBN 978-1-4244-7993-1, ISSN 1522-4880

1.5 Dissertation outline

The contribute of this thesis regards the design and the development of a general framework for medical image registration, called Fuzzy Kernel Regression, and some of its applications. Such dissertation covers several topics, regarding geometrical transformations, similarity metrics, optimizations and interpolation. In addition, the same framework has been succesfully applied to another different problem of image processing: content-aware image resizing. Extensive validation and evaluation are given for the described methodologies.
The thesis is divided as follows: after this indroduction, in Chapter 2 the image registration problem is discussed and its state of the art is presented. In Chapter 3 the Fuzzy Kernel Regression framework is presented from a theoretic point of view, while in Chapter 4 its applications are described in detail. The results of these systems and methods are discussed in Chapter 5 both from a quantitative and a qualitative perspective. In Chapter 6 is shown how the framework can be applied to other problems, in particular the case of content-aware image resizing is presented. Finally, conclusions and future work are stated in Chapter 7.

Chapter 2

State of the art

The problem of Medical Image Registration has been widely studied in the last decades by teams of researchers, and it has been faced with a variety of approaches, both for the general case and for specific images types. However, there not exists a gold standard for such challenge, since the data involved and the purposes followed can be extremely variable. For these reasons a formal problem statement and a catagorization of the image registration problem according to different criteria are given.

2.1 Image Registration problem

The problem of Image registration can be regarded as finding the transformation, generally defined by a set of parameters, that best maps one dataset (namely the input, floating or moving image) onto the other (namely the target, base or fixed image) bringing them in spatial alignment. At the end of the process, corresponding pixels/voxels will have the same positions in both images/volumes. Formally, given a reference image R and a floating image F, the optimal spatial transformation T should be such that applying T to F, the result of the distance operator $E(\cdot)$ (i.e. the error) is minimum (2.1).

$$T^* = \arg\min_{T \in \Gamma} \{E(T)\}. \tag{2.1}$$

In some cases the distance term can be the composition of a data term $E_{diff}(\cdot)$ and a regularization term $E_{reg}(\cdot)$ to prevent the introduction of unwanted transformations. In this case 2.1 becomes as follows:

$$T^* = \arg\min_{T\in\Gamma} \{E(T)\} = \arg\min_{T\in\Gamma} \{E_{dis}(R, F \circ T) + E_{reg}(T)\}. \tag{2.2}$$

The data term can be designed in different ways, generally is an optimization operator of some similarity metrics between the two images.

2.2 Registration classes and taxonomies

Since Image Registration problems span into a broad set of categories, some classifications need to be made. In particular registration algorithms can be divided in four classes:

- *Different viewpoints* (or *multi-view analysis*): The same object or scene is acquired from different viewpoints: the goal is to obtain a larger view or a 3d representation of the object.
- *Different times* (or *multi-temporal analysis*): Images of the same object are acquired in different times, perhaps under different conditions. The aim is to evaluate differences between two or more acquisitions.
- *Different sensors* (or *multimodal analysis*): Images of the same object are acquired by different sensors, for example magnetic resonance (MRI), computer tomography (CT), positron emission tomography (PET), etc.
- *Scene to model registration*: images of a real scene and its model are registered. The model can be a synthetic representation or another scene with similar content. The purpose of this method is to find the acquired image in the model and compare them.

In addition, each method should take into account the objects to be registered and the characteristics of the deformations to be recovered. Furthermore, even more elements as noise corruption should be considered too. Generally, a distinction between feature-based and area-based approaches is operated, depending on how much information is used for the registration task, in the first case just a sparse information subset (the features) is used for recovering the mapping, in the latter all of the image information is taken into account. Nonetheless, every strategy generally uses four steps, with the exception of the first one. These steps are the following:

- *Feature detection*: salient and unambiguous objects such as corners, intersections, contours, etc., are manually or automatically detected in both the input and reference image. This step is omitted in area-based strategies.

- *Feature matching*: the correspondences between the images are found by means of matching the previously detected features. For this purpose, there exist several feature descriptors and similarity measures based on features appearance or informative content. Since area-based strategies use all of the image information; such methods use a dense features map simply defined by all of the pixels/voxels in the images.

- *Transform model estimation*: after the features are matched, this information is used to recover a transformation function, which defines the deformation needed to map every pixel/voxel of the input image onto every pixel/voxel of the reference image. Such a function is determined by choosing its type and defining the value of its set of parameters.

- *Image resampling and transformation*: once the deformation estimation is achieved, the mapping function is applied on the input image. Since this mapping generally brings the pixels/voxels in non-integer coordinates, proper interpolation techniques need to be used in order to avoid or limit resampling artefacts.

Each of these steps has its intrinsic problems, so each one has to be developed taking into account the properties of the objects that have to be registered. For

example the presence of noise can affect feature detection. So, if noise is assumed to be present, the detection procedure should be robust. A potential problem in feature matching is the different appearance of corresponding features due to illumination conditions or to sensors spectral sensitivity; in this case the similarity measure adopted needs to take into account these factors.
For what concerns the transformation function several choices exist (Figure 2.1).

- Rigid
- Affine
- Projective
- Curved

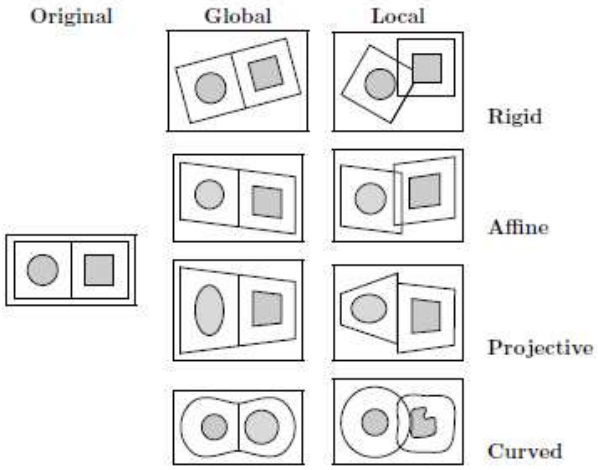


Figure 2.1: Transformation functions used for registration purposes, being applied globally or locally.

Generally, a registration algorithms takes into account the block diagrams defined in Figure 2.2: the floating image is measured against the reference image, if the similarity does not satisfy a stopping criterion, an optimization phase is run in order to tune the transformation parameters, the deformation is applied to the floating image and the cycle is repeated again.

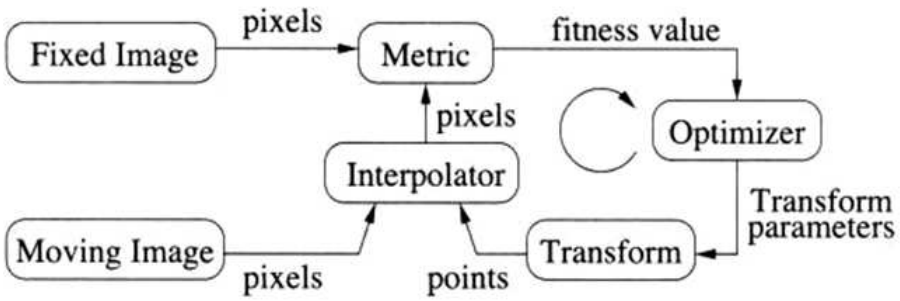


Figure 2.2: Block diagram of the registration steps.

2.3 Current literature

Image registration techniques span into a broad class of methods and taxonomies according to the type of registration basis and the nature of transformation. Several surveys on the subject are present in literature, Brown (1992); Hawkes (1998); Lester & Arridge (1999); Maintz & Viergever (1998); Zitová & Flusser (2003) and as reported in Pluim & Fitzpatrick (2003) this field of research is very active and in growth.

Image registration methods can be landmark-based or area-based. The first type relies on the information provided by some corresponding features into the two images, such as points, lines, regions, etc. The latter type uses the whole image content to estimate the registration transformation by means of optimizing some similarity metric. Among a lot of propsed similarity metrics for such optimization, Mutual Information (MI) and its normalized version (NMI), has proven to be one of the most effective, especially for multi-modality registration tasks, Maes *et al.* (1996); Meyer *et al.* (1997); Studholme (1999); Viola (1995), since it does not assume any functional relationship between the intensity values of the images, but just statistical.

For what concerns the nature of transformation, many models exist in literature. Simplest ones use global or local mapping models using rigid, affine or projective transformations. Others, able to deal with local deformations, use radial basis functions such as Thin-plate spline, Bookstein (1989) or Wendland's functions, Fornefett *et al.* (2001); Wendland (1995). Likar & Pernus (2001) propose a hierachical method based on local rigid or affine transformation. Arsigny *et al.* (2003)

use the so-called polyaffine transformation with differentiability and invertibility properties. Ourselin *et al.* (2000) find block-level correspondences of sub-regions of the images and use this information for recovering a global transformation. Other more complex approaches are to adopt parameters free deformation functions, by considering the image as a tensile material, Bajcsy & Kovačič (1989) or a viscous fluid, Nielsen & Gramkow (1996), deformed by external and internal forces subject to constraints. In this approach, registration is achieved by the iterative minimization of an energy functional.

Using a global method is a practicable choice only when using simple transformation models, where just few parameters are required. When using curved deformations and the number of parameters is large, a direct optimization is not possible, due to large dimensionality of the search space and the presence of many local optima. A possible solution is to decompose the image domain and operate many local sub-image level registrations using simple models. The final global transformation can be recovered by composing the local transformations, obtaining a unique continuous and smooth complex deformation, Gaens *et al.* (1998); Maintz *et al.* (1998).

When using MI or NMI for this purpose, this approach needs particular care because Mutual Information reliability strongly depends on the number of samples used to estimate the joint histogram, so when using small sub-images or large subsampling rates, the result can be compromised by the presence of additional local optima Ji *et al.* (2003); Pluim *et al.* (2000); Tsao (2003). To avoid such problems, a proper interpolation scheme should be used, in conjuction with additional techniques such as oversampling and/or intensity clustering Ji *et al.* (2003).

Chapter 3

Fuzzy Kernel Regression

The registration methods developed rely onto a common framework, designed and developed as part of the research work. Such theoric construct is called Fuzzy Kernel Regression, since it is the combination of two known literature Pattern Recognition techniques: Fuzzy C-means and Kernel regression. The classic kernel regression is enhanced by fuzzy related techniques, in particular the C-means clustering algorithm. In this section an overview of these methods is given. After the explanation it is shown how they are combined to form the proposed framework.

3.1 Kernel Regression

In pattern recognition, there exists a class of techniques, which uses data points or a subset of them not just in the training phase, but also in the prediction phase. These are called memory-based methods. Linear parametric models that can be re-cast into equivalent dual representations where the predictions are given by linear combinations of a kernel function evaluated at the training data points are known as kernel regression methods. Kernel functions are are defined by training data points. Kernels, which depend only on the magnitude of the distance of the argument from the training points, are known as homogeneous kernels or radial basis functions.

For our registration purpose we will use the derivation of kernel regression from the scheme known as the Nadaraya-Watson model (Nadaraya (1964); Watson

(1964)).
Starting from the training set $\{\mathbf{c_n}, \mathbf{t_n}\}$, the joint distribution $p(\mathbf{x}, \mathbf{t})$ can be modeled using a Parzen density estimator:

$$p(\mathbf{x}, \mathbf{t}) = \frac{1}{N} \sum_{n=1}^{N} f\left(\mathbf{x} - \mathbf{c_n}, \mathbf{t} - \mathbf{t_n}\right) \tag{3.1}$$

where $f(\mathbf{x}, \mathbf{t})$ is the component density function. There is an instance of $f(\cdot)$ centered in each sub-image. The regression function $y(\mathbf{x})$, corresponding to the conditional average of the target variable depending on the input, is given by

$$\begin{aligned} y\left(t\right) &= E\left[\mathbf{t}|\mathbf{x}\right] = \textstyle\int_{-\infty}^{+\infty} tp\left(\mathbf{t}|\mathbf{x}\right) dt = \frac{\int tp(\mathbf{x},\mathbf{t})dt}{\int p(\mathbf{x},\mathbf{t})dt} = \\ &= \frac{\sum\limits_{n} \int tf(\mathbf{x}-\mathbf{c}_n,\mathbf{t}-\mathbf{t}_n)dt}{\sum\limits_{m} \int f(\mathbf{x}-\mathbf{c}_m,\mathbf{t}-\mathbf{t}_m)dt}. \end{aligned} \tag{3.2}$$

Assuming that the component density functions have zero mean so that

$$\int_{-\infty}^{+\infty} f\left(\mathbf{x}, \mathbf{t}\right) \mathbf{t} d\mathbf{t} = 0 \tag{3.3}$$

for all values of $\mathbf{x}$, we can operate a variable change, and we get

$$y\left(\mathbf{x}\right) = \frac{\sum\limits_{n} g\left(\mathbf{x} - \mathbf{c}_n\right) \mathbf{t}_n}{\sum\limits_{m} g\left(\mathbf{x} - \mathbf{c}_m\right) \mathbf{t}_m} = \sum_{n} k\left(\mathbf{x}, \mathbf{c}_n\right) \mathbf{t}_n, \tag{3.4}$$

where the kernel function $k(\mathbf{x}, \mathbf{c_n})$ is defined as

$$k\left(\mathbf{x}, \mathbf{c}_n\right) = \frac{g\left(\mathbf{x} - \mathbf{c}_n\right)}{\sum\limits_{m} g\left(\mathbf{x} - \mathbf{c}_m\right)} \tag{3.5}$$

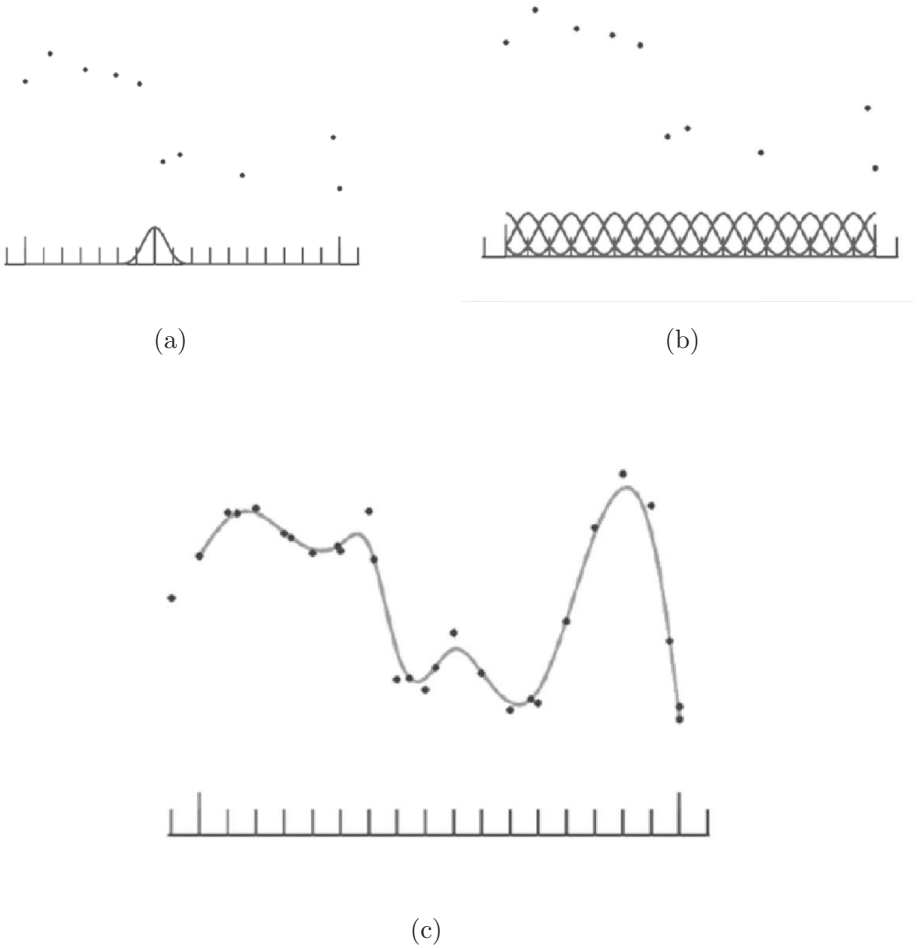
and

$$g\left(\mathbf{x}\right) = \int_{-\infty}^{+\infty} f\left(\mathbf{x}, \mathbf{t}\right) d\mathbf{t}. \tag{3.6}$$

This form is known as the *Nadaraya-Watson* model or *kernel regression*. In case of localized kernel functions, it has the property of weighting more the data points $\mathbf{c_n}$ close to $\mathbf{x}$ than the others. The kernel (3.5) satisfies the summation constraint

$$\sum_{n=1}^{N} k\left(\mathbf{x}, \mathbf{c}_n\right) = 1. \tag{3.7}$$

A graphical representation of a function recovered using kernel regression is reported in Figure 3.1.



(a) (b)

(c)

Figure 3.1: Example of kernel regression functions. Data points and corresponding kernels (a,b), and the reconstructed function (c).

3.1.1 Constructing Kernels

In order to exploit kernel substitution, wa have to construct valid kernel functions. Several approaches exist for this purpose. One consists in choosing a feature space mapping $\phi(x)$ and use it for finding the corresponding kernel. Another possible approach is to construct kernel functions directly. In this case, we have to assure that we choose a valid kernel, i.e. it corresponds to a scalar product in some

feature space. The last (and commonly used approach) is to build new kernels as a composition of simple existing kernels, according to the following rules:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \tag{3.8}$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \tag{3.9}$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \tag{3.10}$$

$$k(\mathbf{x}, \mathbf{x}') = exp(k_1(\mathbf{x}, \mathbf{x}')) \tag{3.11}$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \tag{3.12}$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') \tag{3.13}$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \tag{3.14}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathbf{T}} \mathbf{A} \mathbf{x}' \tag{3.15}$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x_a}, \mathbf{x_b}') + k_b(\mathbf{x_b}, \mathbf{x_b}') \tag{3.16}$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x_a}, \mathbf{x_b}') k_b(\mathbf{x_b}, \mathbf{x_b}') \tag{3.17}$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function form $\mathbf{x}$ to $\Re^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\Re^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x_a}$ and $\mathbf{x_b}$ are valid kernel functions over their respecive spaces.

3.1.2 Kernel regression in n-d

Usually, regression problems can occur for vectorial functions, as in the case of image and volume processing. Kernel regression can be applied to reconstruct n-d vectorial functions by means of being applied to each component separately. At the end of the process, the results are composed to recover the interpolated vectors, Figure 3.2 and Figure 3.3.

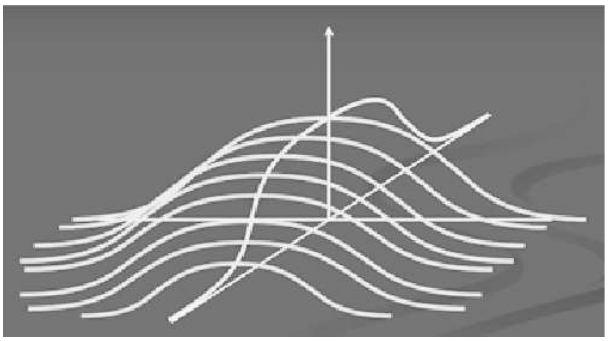
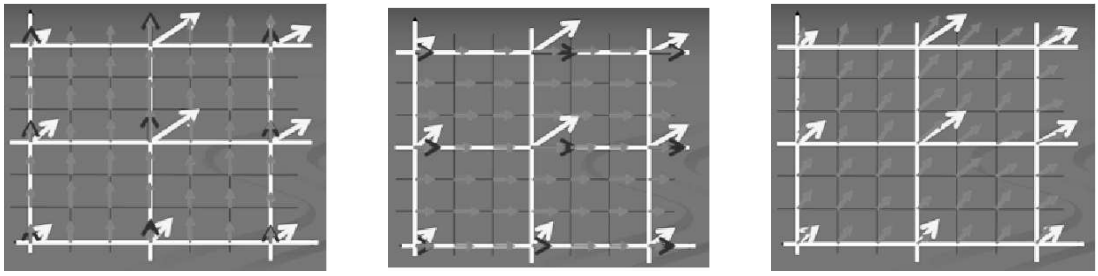


Figure 3.2: Fuzzy C-means example: final membership values assignation and cluster centres positions.



(a) (b) (c)

Figure 3.3: Multi-dimensional regression: components are interpolated individually (a,b) and the results are composed in the final vectors (c).

3.2 Fuzzy C-means

Before explaining how kernel regression can be applied to the registration task, it is necessary to describe the Fuzzy c-means clustering technique, Bezdek (1981), a powerful and efficient data clustering method.
Each data sample, represented by some feature values in a suitable space, is associated to each cluster by assigning a membership degree. Each cluster is identified by its centroid, a special point where the feature values are representative for its own class. The original algorithm is based on the minimization of the following objective function:

$$J_s = \sum_{j=1}^{m} \sum_{i=1}^{k} (u_{ij})^s d\left(\mathbf{x}_i, \mathbf{c}_j\right)^2, \quad 1 \leq s < \infty, \tag{3.18}$$

where $d\left(\mathbf{x_i}, \mathbf{c_j}\right)$ is a distance function between each observation vector $\mathbf{x_j}$ and the cluster centroid $\mathbf{c_j}$, s is a parameter which determines the amount of clustering fuzziness, m is the number of clusters, which should be chosen a priori, k is the number of observations and u_{ij} is the membership degree of the sample $\mathbf{x_i}$ belonging to cluster centroid $\mathbf{c_j}$.
An additional constraint is that the membership degrees should be positive and structured such that $u_{i1}+u_{i2}+\ldots+u_{im} = 1$. The method advances as an iterative procedure where, given the membership matrix $\mathbf{U} = [\mathbf{u_{ij}}]$ of size k by m, the new positions of the centroids are updated as:

$$\frac{\sum_{i=1}^{k} (u_{ij})^s x_i}{\sum_{i=1}^{k} (u_{ij})^s}. \tag{3.19}$$

The algorithm ends after a fixed number of iterations or when the overall variation of the centroids displacements over a single iteration falls below a given threshold. The new membership values are given by the following equation:

$$u_{ij} = \frac{1}{\sum_{l=1}^{m} \left(\frac{d(\mathbf{x}_i, \mathbf{c}_j)}{d(\mathbf{x}_i, \mathbf{c}_l)}\right)^{\frac{2}{s-1}}}. \tag{3.20}$$

To better understand the whole process a one-dimensional example is reported (i.e. each data point is represented by just one value).
Twenty random data points and three clusters are used to initialize the procedure and compute the initial matrix **U**. Note that the cluster starting positions, represented by vertical lines), are randomly chosen. Figure 3.4 shows the membership values for each data point relative to each cluster; their colour is assigned on the basis of the closest cluster to the data point.

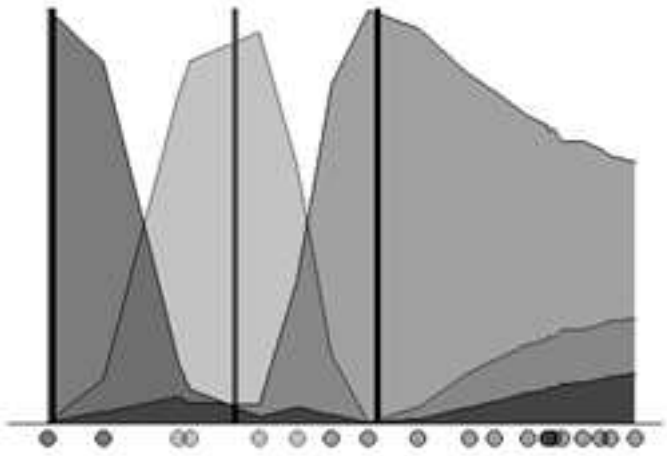


Figure 3.4: Fuzzy C-means example: initial membership values assignation.

After running the algorithm, the minimization is performed and the cluster centroids are shifted, the final membership matrix **U** can be computed. The resulting membership functions are depicted in Figure 3.5

3.3 Fuzzy Kernel Regression

Merging the results of the previous discussion, it turns out that Fuzzy C-means membership functions can be used as kernels for regression in the Nadaraya-Watson model since they satisfy the summation constraint (3.7). In the scenario of image registration, the input variables populate the feature space by means of the spatial coordinates of the pixels/voxels and cluster centroids are represented

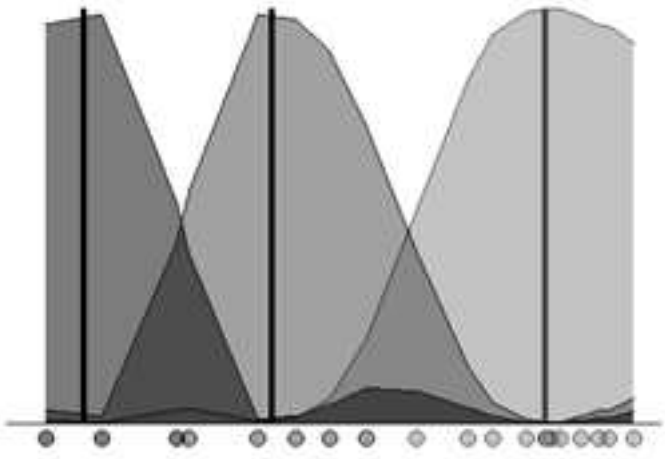


Figure 3.5: Fuzzy C-means example: final membership value assignation and cluster centres positions.

by relevant points in the images, whose spatial displacement is known. For example, landmark points where correspondences are known between input and reference image can be used for this purpose.
As a result of such setting there is no need to execute any minimization of the Bezdek functional, since image points are already supposed to be clustered around the landmark points (or equivalent representative points). Fuzzy C-means is used just as a starting point for the registration procedure. Once the relevant points are known, a single FCM step is performed to construct Fuzzy kernels by means of computing membership functions. For this purpose the distance measure used in (3.20) is the simple Euclidean distance, since just spatial closeness is required to determine how much any point is influenced by surrounding relevant points. Such membership functions are then used to recover the displacement for any pixel/voxel in the image using the following formula:

$$y\left(\mathbf{x}\right) = \sum_{n} u\left(\mathbf{x}, \mathbf{x_n}\right) t_n, \tag{3.21}$$

where $u(\mathbf{x}, \mathbf{x_n})$ is the membership value for the current pixel/voxel with re-

gard to the relevant point xn, and tn is a 2d/3d vector or function representing its known xy or xyz displacement. This will result in continuous and smooth displacement surfaces, which interpolate relevant points.
Even if the registration framework is unique, it can be applied in several ways, depending on the choice of the target variable, i.e. what is assumed to be the prior information in terms of relevant points and their known displacement. In the following chapter three different applications of the proposed framework will be described.

Chapter 4

Proposed systems

In the previous chapter the theory of the registration framework designed for the registration purpose has been descripted. In this chapter an exaplanation of the actual registration systems implemented will be given. Baiscally, three applications have been developed. Beside increased complexity, the main difference between them lies in what are considered to be the target variables t_n and how prior information is obtained. In addition to the algorithmic details, further attention is given to interpolation scheme. As will be shown, such issue is critical in image registration, in particular for Mutual Information maximization based procedures. Finally implementation details taken into account for tuning, optimizing and improving the overall efficiency of the systems will be explained.

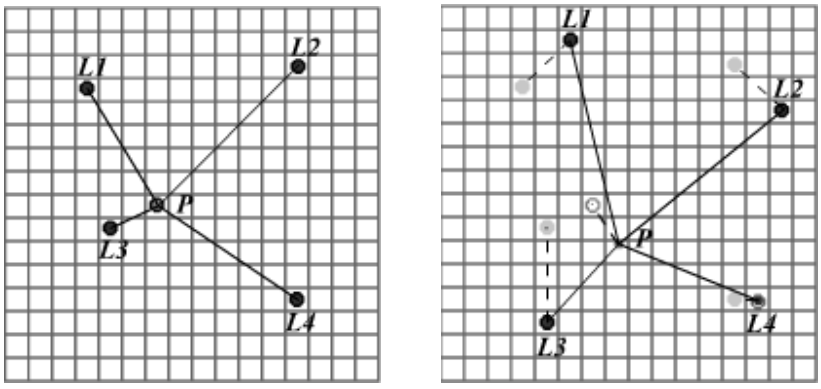
4.1 Simple landmark based registration

A first application arises naturally from the described framework. It is very simple and is meant to demonstrate the actual use of the fuzzy kernel regression. However since it is effective notwithstanding its simplicity, it could be used for actual registration tasks.
Basically, it consists in considering the landmark points themselves directly as the relevant points representing the cluster centroids for the FCM step, and their displacements vectors directly as the target variables. Each pixel/voxel is then subjected to a displacement contribute from each landmark point. Such contribute is high for closer points and gets smaller while relative distances between

the input points and the landmarks increase. The final displacement vector for any input point will consequently be a weighted sum of the landmarks points movements.
To better understand this technique an example of the procedure is explained: a pattern image showing four landmark points is depicted in Figure 4.1a. An input point P is considered, and its distances from the four landmarks are shown. After the procedure is applied with a fuzziness value s set to 1.6, the point P results to have the following membership values for the four landmarks:

$$y(\mathbf{x}) = \sum_n u(\mathbf{x}, \mathbf{x_n})\, t_n, \tag{4.1}$$

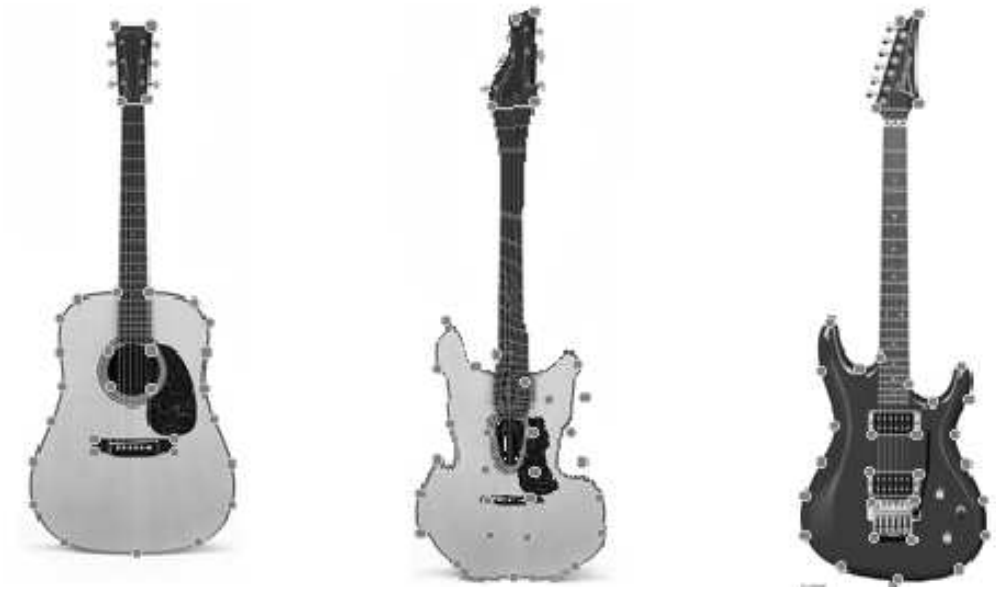
This means that it will receive the greatest part of the displacement contribute from the bottom-left landmark, and just a marginal contribute from the other three. The results are confirmed in 4.1b, where the point has been moved according to a displacement vector that is mostly similar to the displacement of the third landmark. Anyway, other landmarks give small influences too.



(a) (b)

Figure 4.1: Example of single point registration using four landmarks.

Repeating the same procedure for the points in the whole image, complete dense displacement surfaces are recovered, one for each spatial dimension. Such surfaces have continuity and smoothness properties.
As a first example, visual results for conventional images are shown in Figure 4.2.



(a) (b) (c)

Figure 4.2: Example of registration of conventional images. Input image (a), registered image (b) and target image (c). In this example 31 landmark points were used with the fuzziness s value set to 1.6.

In Figure 4.3 are shown the recovered displacement surfaces for x, (Figure 4.3a) and y (Figure 4.3b) values respectively.

4.2 Improved landmark based registration

Although the simple method previously described is effective and can be useful for simple registration tasks, it does not result suitable for many applications in that it does not take properly into account relations between neighbouring landmark points. In other words, considering a single point displacement vector to represent the deformation of the image in different areas is not enough. Thus, it is necessary to find an effective way of estimate such zones. Given some landmark points, a simple way to subdivide the image space in regions is the application of the classic Delaunay triangulation procedure, Delaunay (1934), which is the optimal way of

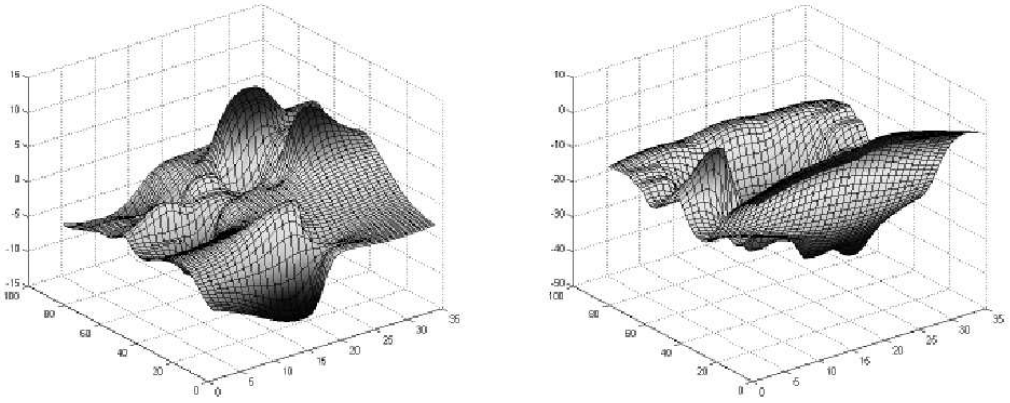


Figure 4.3: Displacement surfaces recovered for x (a) and y (b) values.

recovering a tessellation of triangles, starting from a set of vertices. It is optimal in the sense that it maximizes the minimum angle among all of the triangles in the generated triangulation. Starting from the landmark points and their correspondences, such triangulation produces a most useful triangles set along their relative vertices correspondences. An example of Delaunay triangulation is depicted in Figure 4.4.

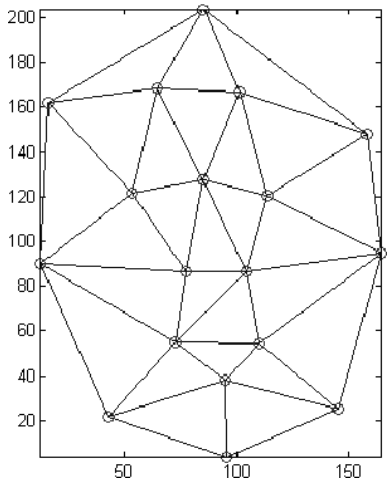


Figure 4.4: Example of Delaunay triangulation.

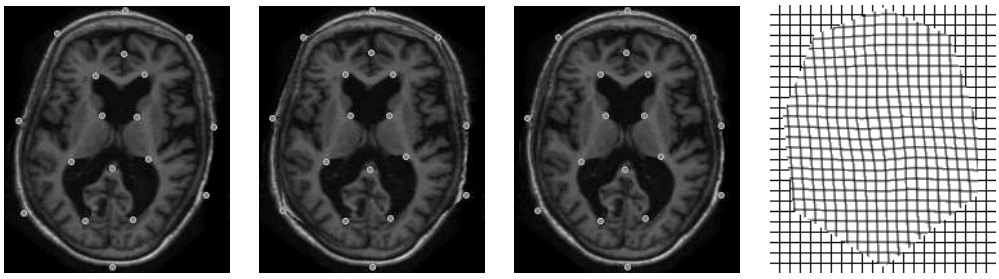
Once we have such triangle tessellation whose vertices are known as well as

their displacements, it is possible to recover the local transformations, which maps each triangle of the input image onto its respective counterpart in the target image. Such transformation can be recovered in several ways; basically an affine transformation can be used. In 2d space affine transforms are determined by six parameters. Writing down the transformation equation (4.2) for three points a linear system of six equations to recover such parameters can be obtained. Similar considerations hold for the three-dimensional case.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} ax_0 + by_0 + c \\ dx_0 + ey_0 + f \\ 1 \end{bmatrix} \Rightarrow \begin{cases} x_1 = ax_{0,1} + by_{0,1} + c \\ y_1 = dx_{0,1} + ey_{0,1} + f \\ x_2 = ax_{0,2} + by_{0,2} + c \\ y_2 = dx_{0,2} + ey_{0,2} + f \\ x_3 = ax_{0,3} + by_{0,3} + c \\ y_3 = dx_{0,3} + ey_{0,3} + f \end{cases} \tag{4.2}$$

Each transformation is recovered from a triangle pair correspondence, and the composition of all the transformations allows the full reconstruction of the image. Anyway, this direct composition it is not sufficient per se, since it presents crisp edges because transition between two different areas of the image are not smooth even if the recovered displacement surfaces are continuous due to the adjacency of the triangles edges. This can lead to severe artefacts in the registered image, especially for points outside of the convex hull of the control points (Figure 4.5c and Figure 4.5d), where no transformation information are determined. To better understand this problem an example of registration along the recovered surfaces plot are shown respectively in Figure 4.5 and Figure 4.6.

Fuzzy kernel regression technique can be used to overcome this drawback. To apply the method, relevant points acting as cluster centroids must be chosen. Since our prior displacement information is no more about landmark points, but about triangles, they cannot be chosen as relevant points anymore. Thus, we have to choose some other representative points for each triangle. For this purpose, centres of mass are used as relevant points, and their relative triangle affine transformation matrix is the target variable. In this way, after recovering the membership functions and using them as kernels for regression, final displacement for each pixel/voxel is given by the weighted sum of the displacements given by all of the affine matrices. In this way the whole image information is taken into



(a) (b) (c) (d)

Figure 4.5: Example of registration of conventional images. Input image (a), registered image (b) and target image (c). In this example 31 landmark points were used with the fuzziness s value set to 1.6.

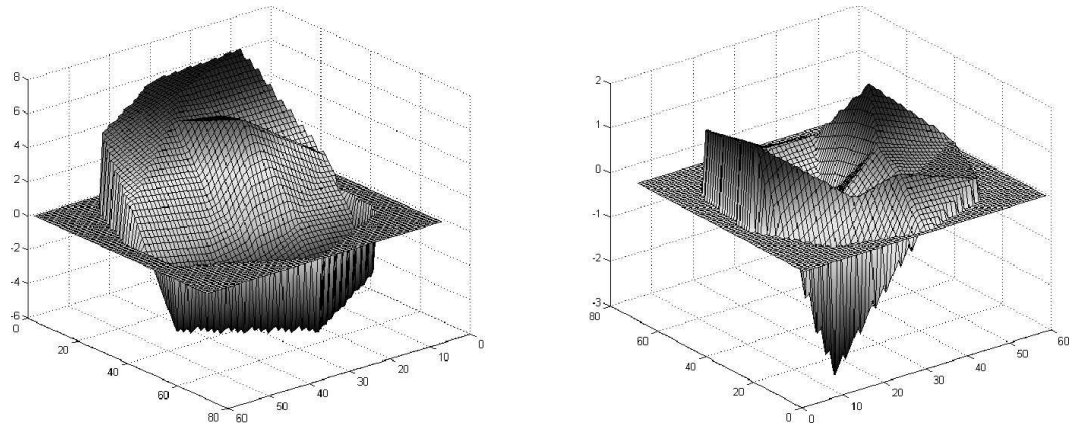


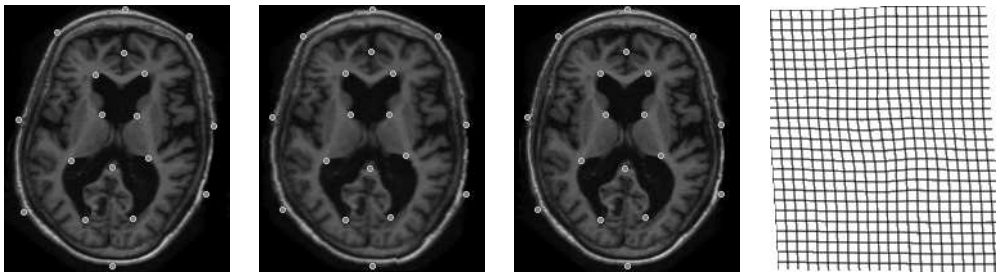
Figure 4.6: Displacement surfaces recovered for x (a) and y (b) values with direct affine transformation composition.

account. The final location of each pixel/voxel is then obtained as follows (2d case):

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \sum_n \left(u_n(x,y) \begin{bmatrix} a_n & b_n & c_n \\ d_n & e_n & f_n \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \right). \tag{4.3}$$

In this way there are no more displacement values that change sharply when crossing triangle edges, but variations are smooth according to the choice of the fuzziness parameter s. In Figure 4.7. and Figure 4.8 registration results and

deformation surfaces for the previous examples are shown. Note that there are no more sharp edges in the surface plots and a displacement value is recovered also outside of the convex hull defined by the landmarks points.



(a) (b) (c) (d)

Figure 4.7: Example of MRI image registration with fuzzy kernel regression affine transformations composition. Input image (a), registered image (b) and target image (c). Deformed grid in (d). In this example 18 landmark points were used.

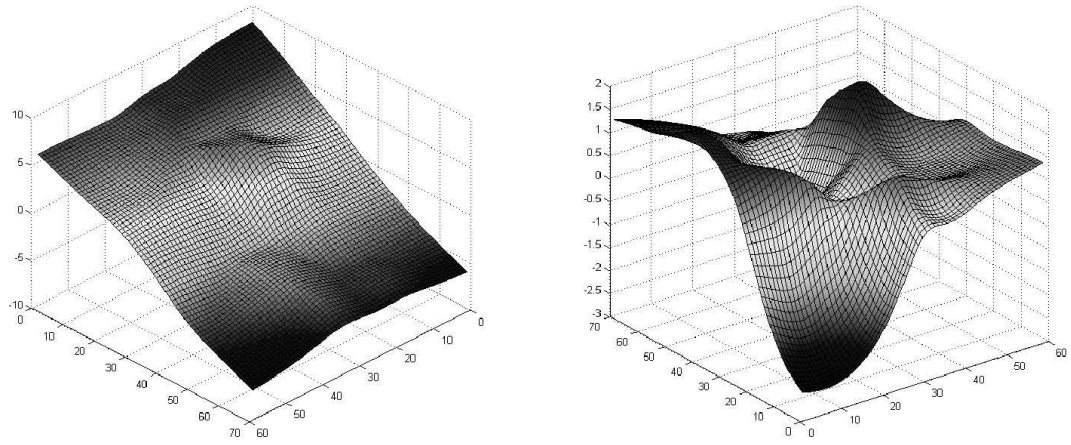


Figure 4.8: Displacement surfaces recovered for x (a) and y (b) values with fuzzy kernel regression affine transformation composition.

4.3 Automatic intensity based registration

Differently from the previous methods, last one is area-based, no landmark points need to be selected, since correspondences are determined during the registration process itself. In this case the problem of aligning input image I and target image T can be represented as the problem of finding the optimal spatial transformation F^* able to match I and T. Such parameters are found by minimizing a cost function E:

$$F^* = \arg\min_{F \in \Gamma} \{E(F)\} = \arg\min_{F \in \Gamma} \left(E_{dis}(T, I \circ F) + E_{reg}(F)\right) \tag{4.4}$$

where the set Γ is the space of all the admissible transformations. The term $I \circ F$ represents the transformation of I subject to F. E is divided into two parts: the data dissimilarity term E_{dis} and E_{reg}, which is the optional regularization term, used to penalize undesired transformations. E_{dis} can be designed using several form and functionals: in our work we adopted Normalized Mutual Information (NMI). Therefore the data term become as follows:

$$E_{dis}(T, I \circ F) = Y(T, I \circ F) = \frac{H(T) + H(I \circ F)}{H(T, I \circ F)}, \tag{4.5}$$

where $H(T)$ and $H(I \circ F)$ are the entropies of the target image and of the transformed input image respectively, and $H(T, I \circ F)$ is their joint entropy. Since in our method registration is piece-wise, the minimization of the cost function is performed separetely in each sub-region considered in order to recover local registration transformations. The composition of such deformations is then operated by means of fuzzy kernel regression. The complete registration algorithm is realized by several steps which are summarized in the block diagram reported in Figure 4.9.

Since mutual information is sensitive to noise, a preproccesing step aimed to noise reduction is applied to mitigate this weakness: a binary mask is used to separate the content from the background, which is cutted and discarded. The second step consists in the application of a global affine transformation used a starting point for the successive elastic registration, as in Kohlrausch *et al.* (2005). Such strategy reduces large misalignments and provides a speed up for the convergence of the succissive steps. The core of the method is the elastic registration.

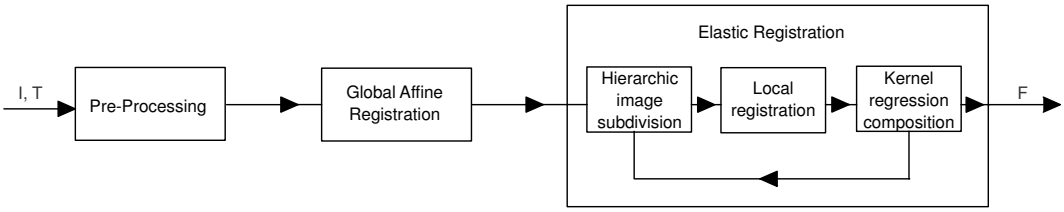


Figure 4.9: Blocks diagram for the area-based fuzzy kernel regression registration algorithm. First block performs a binary image pre-processing. Second step computes a first coarse global affine registration used as starting condition for the elastic registration procedure, which performes iteratively three steps. Firstly a hierarchical image subdivision is operated, then the resulting subimages are aligned locally, finally a smooth composition of the registered image is achieved by means of fuzzy kernel regression.

This step is realized with the smooth composition of several local affine transformations. In particular, such transformations are evaluated hierarchically, form a coarser to a finer level of detail, i.e., the extent of the region subject to the effect of the computed transformations, gets smaller and smaller throughout the evolution of the procedure. This allows to take into account and align finer details. For this purpose the input image is subdivided into several sub-images. Then, for each sub-image, the local optimum transformation aligning it to the the target image is computed maximizing the NMI measure. The recovered transformations are then propagated in the whole image using a composition based on the fuzzy kernel regression model. At each step the input image is subdivided with a regular grid of variable size depending on the current level of detail considered. The centres of the resulting regions are the cluster centroids $\mathbf{c}_n$. Starting from this disposition the fuzzy membership map $\mathbf{U}$ can be recovered. Note that it needs to be evaluated only once for each level of detail and can be evaluated as a lookup table during the iterations at a certain resolution. Each local affine transformation matrix recovered from each sub-region alignment represents the target variable $\mathbf{t}_n$. Once this values are known local deformations can be composed as in the improved landmark-based approach using (4.3). As a consequence of the whole process, each pixel in the image will be subject to a motion vector whose direction and intensity are influenced, with the proper extent, by all of the local transformations recovered for each sub-image. The closest regions will influence

the vector at the maximum degree, while furthest ones will provide progressively minor contributions. The resulting deformation surface will be continuous and smooth. As for the other methods, the amount of smoothness is governed by the fuzziness tunable value s.

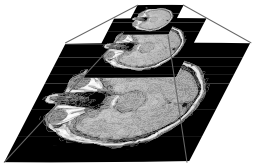


Figure 4.10: Pyramidal structure for the elastic registration step. The similarities are evaluated and optimized at different level of detail (8, 4 and 2), from coarser to finer, in order to speed up convergence and save computational time.

4.3.1 3d extension

The procedure was described as a 2d method. However, the concepts of fuzzy kernel regression, hierarchical decomposition and local registration can be used in 3d as well, without loss of generality. The main differences are:

- The image is no more decomposed into rectangular regions, but in box-shaped regions.
- Fuzzy kernel regression is used to recover three deformation surfaces, respectively for x, y and z dimensions.
- Local affine transformation matrices require the estimation of twelve parameters instead of six.

In Figure 4.11 is shown the hierarchical approach used for 3d registration, where, as level of detail increases, image volume is partitioned into progressively smaller subvolumes pairs which are registered by means of affine transformation matrices parametrized with the maximization of NMI. After each volumes pair is registered the resulting pieces are composed using fuzzy kernel regression to obtain a unique transformed volume.

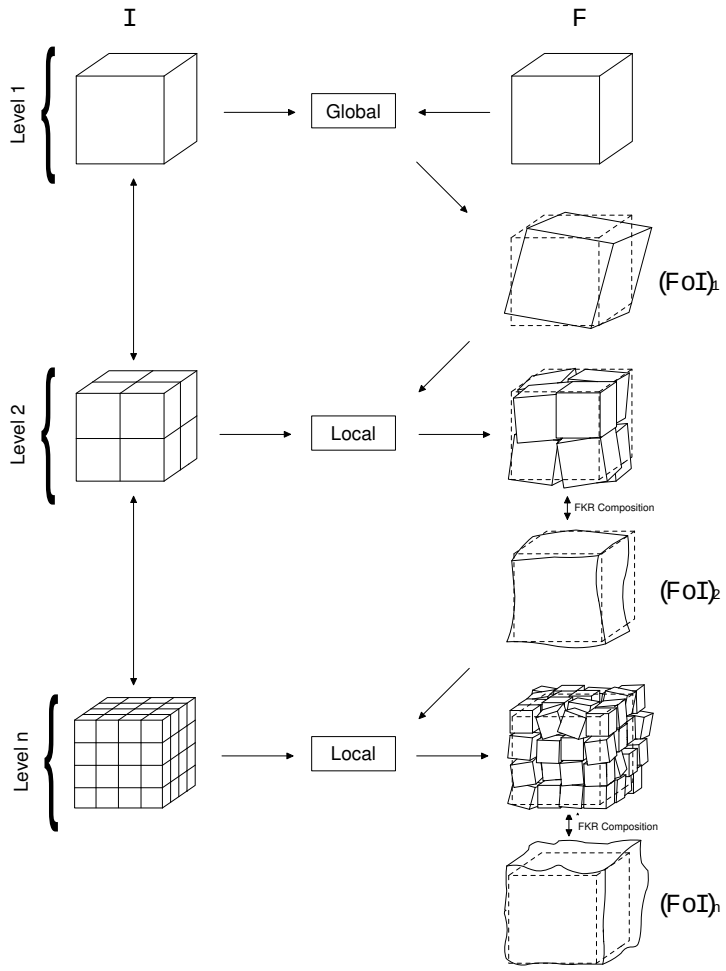


Figure 4.11: Hierarchical approach to elastic image registration.

4.4 Interpolation

During the process of similarity metric optimization (like the one involved in automatic intensity based registration), and especially for Mutual Information maximization (as proved in several works, Ji *et al.* (1999); Pluim *et al.* (2000); Tsao (1999)), a fundamental issue is interpolation. The importance of adopting a good interpolation scheme regards two aspetcs: local optima reduction and image reconstruction. While the latter is just an image quality issue, the first is critical for the performance of a registration system.

Mutual information is sensitive to artifacts introduced due to the interpolation

needed to apply the geometric transformations. These errors can introduce local maxima in the mutual information function. Such local maxima prevent the optimizer from finding the actual solution, thus not providing the best datasets alignment.

Typical interpolators recover the image intensities in non-integer coordinate points, where the values are unknown, by convolving a kernel function with the image. This procedure is analog to a resampling step. The larger is the support of the used interpolator, the smaller is the resampling error introduced. However, the best image reconstruction, does not necessarily translate to a better registration. This can be the result of several causes. One of these is the rounding error due to the binning process required in the joint histogram estimation. In addition, some other factors such as image noise can produce severe artifacts using standard interpolators. It is important to remark that for registration purposes what is important is not how high is the value of the maximum found, but rather its position in the admissible transformations space. This means that probably two different methods should be used for accomplishing, separately, the two tasks.

In order to solve this issue, a novel interpolation scheme was designed. It is based on simple distance measures so it keeps affordable for computation purposes even for processing large amount of data. The method uses optimized operators to achieve maximum efficiency. A variety of tests have been conducted to validate such scheme from both effectiveness (i.e. reduction of local maxima in MI) and efficiency (i.e. computation efforts required) points of view. Experimental results were compared with some literature interpolators: 1) Nearest Neighbor (NN), 2) Linear (LI), 3) NN with jittered sampling (NNJIT) and 4) Partial Volume Interpolation (PVI) Maes *et al.* (1997). Results show that the method provides a good tradeoff between effectiveness and efficiency for registration purposes. In addition it provides good image reconstruction too, representing a good unique candidate for both tasks. Some prior studies on this matter have used rigid transformation such as pure translation or rotation Ji *et al.* (2003); Pluim *et al.* (2000); Tsao (2003). For method evaluation non-rigid transformations were used instead.

4.4.1 Interpolation effect

When applying a registration algorithm to a pair of images the main problem is that for several reasons the MI function does not result to be a concave function, and significative fluctuations leading to local maxima are present. The main causes of such fluctutions are imputable to interpolation effects and local similarities in the images. However, since the latter is a problem that exists no matter what the adopted similarity metric is, we will cope just with the interpolation problem.
In Ji *et al.* (2003) is shown that the interpolation problem will always be present even using an ideal interpolator. This results as a direct consequence of the sampling process. Thus, what is possible to do is just to reduce the interpolation effects by using strategies such oversampling or intensity clustering, coupled with an effective interpolation method. In addition it is worth noting that an effective interpolator from a visual persepctive still can have undesirable effects in the MI metric.
In general different interpolators yield different, sometimes dual, kind of artifacts.

4.4.2 Literature interpolation schemes

During registration process interpolation is needed, after any transformation, to recover voxels intensities from non-integer coordinates. For this purpose several interpolation techniques can be used. However, since this operation is extensively repeated during the procedure, a good tradeoff must be chosen between speed and precision.
The quickest though simplest interpolator is nearest neighbor (NN), which assigns the intensity value of the closest voxel. Although this choice can be sufficient for some applications, it is not suited in cases of sub-voxel accuracy or large magnifications, thus resulting not convenient for elastic registration where generally both of them are locally present. In addition it causes severe artifacts in the MI metric.
A most used approach is linear interpolation (LI), which weighs linearly, along each direction, the intensity contribute of all the six voxels in the neighborhood proportionally to their distance from the considered point. It gives better results than NN even though it produces local maxima in the MI Pluim *et al.* (2000).
Cubic interpolator (CI) provides better results than LI but is more expensive

from a computation cost perspective. In theory the best and ideal, though most complex, interpolator is provided by the *sinc* kernel. However, it cannot be reproduced in practice due to its infinite support extent, so various approximations have been proposed, from direct truncation to more complex windowing, some of the most used are Hamming's raised cosine or Lanczos Blinn (1998) windows. Other strategies (NNJIT) can enhance standard interpolation schemes making use of a normal distributed random offset in order to effectively reduce grid-dependent artifacts (which do not often occur in elastic deformations) but on the other hand this introduces small stochastic perturbations. Another practice is to use joint histogram blurring (BLUR) which results in MI curve smoothing. Both NNJIT and BLUR can be used together to overcome the respective problems at an additional computing cost. Last kernel based interpolator is Gaussian filtering which gives results similar to cubic interpolator. One non-conventional approach commonly used in the latest years is PVI Maes *et al.* (1997) which weighs voxels in neighborhood proportionally to the volume defined by their locations and the requested voxel at non-integer coordinates. Although this method is efficient it provides severe artifacts in pure translational transformations Ji *et al.* (2003).

4.5 Mutual Information Remarks

Mutual Information is an information theory concept which expresses the amount of information that a variable A contains about a variable B. It can be expressed in several ways, all of them equivalent. One of the most common form is the following Cover & Thomas (1991):

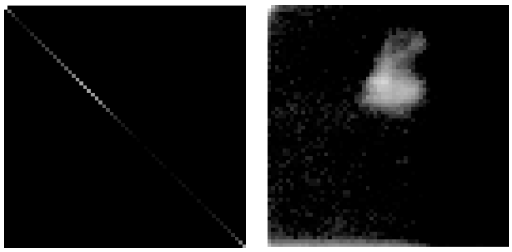
$$I(A,B) = \sum_{S_A}\sum_{S_B} p_{AB}(i,j) \log \frac{p_{AB}(i,j)}{p_{A(i)}p_{B(j)}}. \tag{4.6}$$

This equation states that the distance between the joint distributions of the gray levels contained in the images in case of dependency $p_{AB}(i,j)$, and in case of independency $p_{A(i)}p_{B(j)}$ is a measure of the relative dependency between the two images. Such dependence between pixels' gray levels is maximum when the images are aligned. On the other hand, when this measure decreases a misregistration between the two images exists.
Mutual information has the following properties:

1) $I(A, B) = I(B, A)$: since it is mutual information, the amount of information provided by one image toward the other one is symmetric. However, this is true only in theory, since finite machine precision and interpolation error can introduce differences.
2) $I(A, A) = H(A)$: The self information of an image is equal to its own entropy.
3) $I(A, B) \leq H(A), I(A, B) \leq H(B)$: Mutual information cannot be greater than the information contained in the images themselves.
4) $I(A, B) \geq 0$: Acquiring information about B cannot increase the uncertainty about A.
5) $I(A, B) = 0$ if and only if A and B are independent. In this case acquiring information about B does not add any knowledge about A.
In practice, having some evidence about A and B, there are several ways of estimating the required probabilities for the computation of $I(A, B)$, such as histogram-based methods Maes *et al.* (1996); Moddemeijer (1989), kernel-based methods (for example Parzen windows) Viola (1995); Wells *et al.* (1996), and other parametric models. However, we use histogram-based method since it is one of the most used methods for image registration purposes.
The joint histogram of two images is a bidimensional histogram where each of the axis represents the intensity levels of the two images. In this way, the value of each cell in the histogram is incremented each time a pair $I1(x, y), I2(x, y)$ occurs in the images. If the two images are identical, the result is a population of values disposed only along the main diagonal of the histogram (see. Figure 4.12a), otherwise, if differences or misalignments are present, a dispersion of the values will occur (Figure 4.12b). However, as happens for multimodal medical images, if different gray levels correspond to aligned anatomical structures, the values in the histogram will cluster into regions.

4.5.1 Proposed interpolation scheme

From the study conducted in Pluim *et al.* (2000) it resulted that polynomial and PV interpolation suffer from opposite problems. Polynomial interpolation generates new intensity levels when images grid get unaligned, while PVI generates peaks in the MI function when grids overlap. For this reason, the ideal choice for a new interpolator would be to design a function of the neighborhood which



(a) (b)

Figure 4.12: Left: joint histogram of two identical images. Right: joint histogram of two unaligned images.

eliminates the disadvantages of the two techniques while preserving their advantages.
This needs bring to the construction of an interpolation function which linearly depends by the distance of the neighbors in its computation but using a strategy similar to PVI Maes *et al.* (1997) in order not to introduce new spurious intensity levels.
The neighborhood is represented by the pixels/voxels surrounding the considered point. The idea is very simple: we choose to use the same schema of PVI but the weights w_n are not given by the volumes defined by the neighbors and the requested voxel, but by their euclidean distances. A 2d diagram of the approach is shown in Figure 4.13. Resulting values are then normalized to sum to 1:

$$w_n = \frac{d_n}{\sum_{m \in N} d_m}, \tag{4.7}$$

where N is the set of the neighboring pixels/voxels.

4.6 Implementation issues

A huge amount of computation is required for the solution of the image registration problem, so particular care should be taken for keeping the process affordable and efficient. To speed up convergence, as mentioned, a multi-resolution pyramidal approach has been used. A three level pyramid has been built from input and target images. At each level the images are subsampled with different factors,

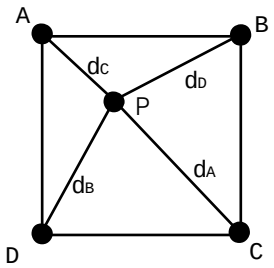


Figure 4.13: Illustration of the proposed interpolation schema: each neighbor's intensity level is weighted by the opposite euclidean distance.

which are 8, 4 and 2. The deformation is evaluated iteratively at each level. The deepest is the level, the finer are the details that can be taken into account.
However, such scheme per se is not sufficient to achieve affordable performances, in particular for the 3d case. For this reason, additional improvements are needed to accelerate the process. Many operations, notwithstanding their simplicity are repeated thousands to millions times, so even a small improvement in a little computational detail can provide a substantial speed up. Two types of improvements have been introduced, code parallelizations and code optimizations. These will be explained in the following paragraphs.

4.6.1 Parallelization with CUDA and GPU clusters

In the last years, GPUs (Graphical Processing Units) gained a large diffusion, firstly for graphics-based applications, and successively for general purpose computations. For this reason the new acronym GPGPU (general purposes GPU) was introduced. This technology can be briefly described as the combination between hardware and software which allows to use traditional GPUs for any type of computation. For a detailed explanation about GPU computing, refer to Appendix A.
Many vendors, such as nVidia and ATI have designed and developed branded solutions for GPUs and a standard API called OpenCL has been proposed as a common layer to provide code compatibility between them. However, each vendor provide its own API for developing onto their cards.
Since it is a common opinion that OpenCL drivers are not mature enough, it was chosen to use specific hardware and API from a single vendor. After a research,

and thanks to the donation received, nVidia and CUDA were chosen as development platform. Detailed reference about CUDA can be found in Appendix A. Basically, the following three operations have been parallelized in the system (which, for what concerns the rest of the dissertation is AAB):

- *Joint histogram generation*: used for MI computation, represents the co-occurence matrix of the intensity level into the two images considered.
- *Fuzzy c-means algorithm*: needed for speeding up the fuzzy maps generation used for kernel regression.

4.6.1.1 Joint histogram generation

Parallel joint histogram generation is an issue which, although in appearance is an extremely good candidate for parallelization, is hard to implement effeciently. The operation consists in incrementing by one the histogram bin located at the coordinates defined by the intensity values of the floating and reference images. Although this is in theory an embarassingly parallel operation, a problem occurs. If the histogram updates are made in parallel synchronization issues known as race conditions occur. Parallelizing a histogram with B bins over N threads is schematically shown in 4.14. Updates to the histogram memory is data dependent, this can results in race conditions and memory access conflicts. For this reason one of the main problem concerns the resolution of such conflicts.
A typical solution, for medium sized histograms, is to produce several sub-histograms with conflicts-free access, and then combine them into the final histogram. For a joint histogram of size *256x256x4 byte = 256 kB*, this is possible. However, this size is larger than the actual shared memory size, so the histogram computation has to be segmented too. The computational schema proposed is shown in 4.15. Each dataset is splitted into partitions which are delegated to each thread block (three in the example), which for each step compute a sub-histogram for a single segment. Before proceeding with the next segment, current one is updated to global memory. Using compute capability 2.0+ (*Fermi* architecture), such steps can be performed in parallel using *streams*). The last consideration is for global memory update. Since CUDA provides atomic operations from compute capability 1.1, not all of the GPUs allow to use them. Then, for old generations GPUs, the approach is to make each thread block update each

sub-histogram segment into different global memory location, and subsequently perform a reduction to the complete histogram with $log(n)$ steps where n is the number of sub-histograms.

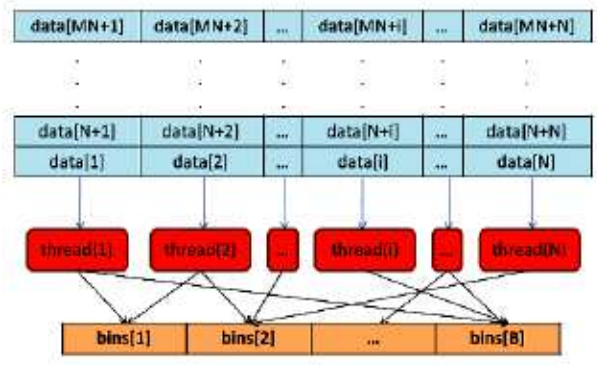


Figure 4.14: Scheme for the parallel calculation of an histogram with B bins distributed over N threads. Update conflicts make necessary the synchronization of the threads to the device memory containing the histogram.

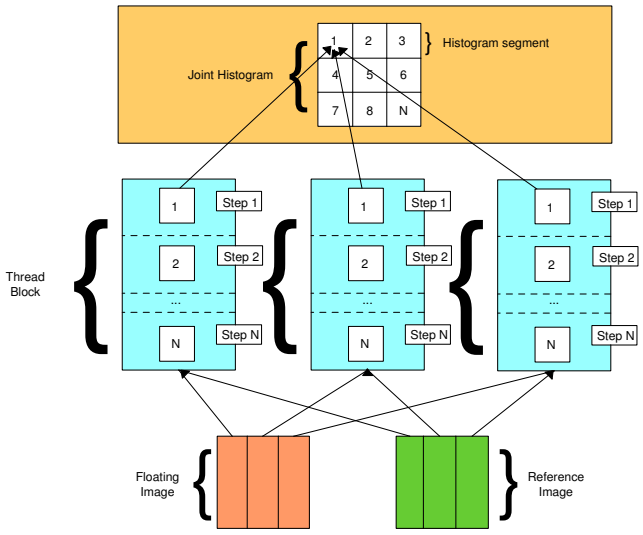


Figure 4.15: Scheme for Joint Histogram computation over 3 thread blocks. Each block computes a segment of its own sub-histogram. At the end of each time step the result is update into the global memory to obtain the complete segment.

4.6.1.2 Fuzzy c-means algorithm

Fuzzy c-means clustering process consists essentially in the manipulation of the data matrices containing the data points and cluster centres. This can be efficiently accomplished using GPUs since there is low data dependency and high parallelization rates can be achieved. However, in literature, only one work facing this issue exists, but it uses traditional GPUs dealing with the graphic pipeline, Anderson *et al.* (2008). In this work, GPGPUs where used, allowing a more flexible management of data structures and algorithms (for example there is no need of using textures for managing arrays).
The computation is spread across several kernels, each one performing some processing. The six-pass procedure for FCM is shown in Figure 4.16.

Kernel 1 computes the (euclidean) distance matrix **D** from **X** and **C**. Kernel 2 takes **D** and computes the new membership values **M**. Since for the rest of the computation membership values are used raised to sth power to avoid recomputations. Kernel 3 multiplies the membersip values for the data points, creating the individual terms of the centers update equations. Kernels 4 and 5 operate two reductions to obtain the summation of the values in the centers update equations, which are finally divided by kernel 6. Reduction is an operation repeated over a series of elements to produce a final scalar values. Examples of reductions are the *sum*, *min* and *max* operators. A parallel reduction algorithms takes $log_2(N)$ number of passes, since at each time step is processed a fraction of the previous timestep results. The first pass processes $N/2$ elements, the second $N/4$, and so forth. A graphical example of reduction is shown in Figure 4.17

In Table 4.1 and Figure 4.18 is reported a plot showing the speedup of GPU versus CPU fuzzy c-means clustering. This example reports the results using 4096 data points, 64 clusters and varying the feature space dimensionality between 8 and 128. Tests were performed on a Nvidia Tesla C2070 GPU. The outcomes point out that the performance increases linearly as the feature space dimensionality grows. Thus, the speed up becomes higher as the feature space becomes larger. Finally, in Table 4.2 absolute performances for various large and very large clustering profiles are reported.

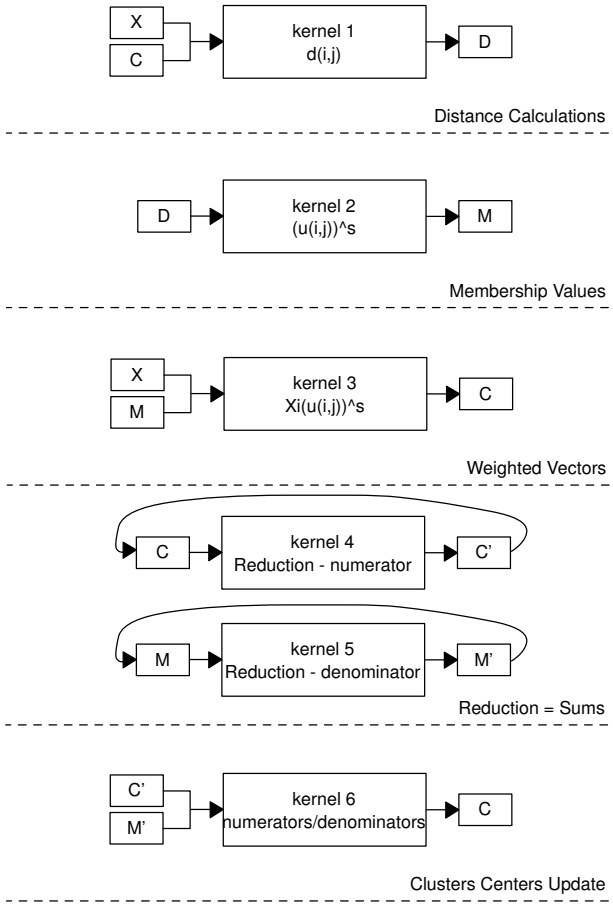


Figure 4.16: GPU algorithm for fuzzy clustering. **X** is the dataset, **C** are the cluster centers, **M** are the membership values and **D** the distance matrix. Kernel 1 computes the distance matrix, kernel 2 updates the memebrship values, kernel 3 computes the numerator for centers update, kernels 4 and 5 operate the reduction of the numerator and denominator of the centers update equation, and kernel 6 accomplish the centers update.

4.6.2 Additional improvements

Some additional improvements have been introduced to make the computation faster. Particular care has been devoted to interpolation. Since this operation will be extensively repeated throughout the whole registration process, it should be implemented in the most efficient way possible. To achieve this purpose and save computational time, most of the effort should be done with the most complex

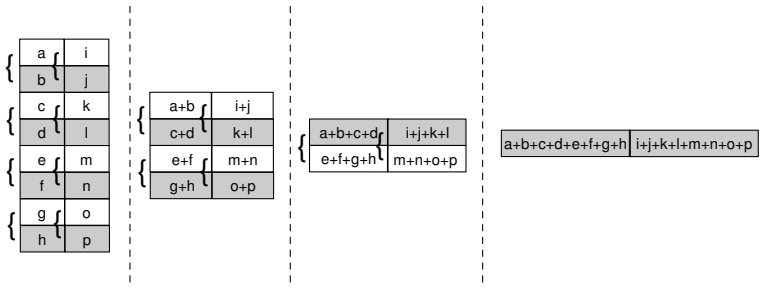


Figure 4.17: Parallel reduction scheme. At each iteration, elements are reduced to an half.

Table 4.1: Speedup measured as CPU/GPU ratio of fuzzy c-means clustering using 4096 data points, 64 cluster varying the feature space dimensionality.

Dimensionality	**CPU/GPU ratio**
8	29,852
16	44,216
24	50,101
32	42,600
40	49,984
48	56,525
56	63,742
64	65,978
72	73,847
80	75,266
88	78,215
96	85,488
104	80,946
112	78,585
120	92,352
128	104,445

function involved in the process, that is square root function. For its evaluation we chose to adopt Newton's method, such procedure computes iteratively the square root as:

$$\sqrt{b} \approx x_{t+1} = 0.5 \left(x_t + \frac{b}{x_t} \right). \tag{4.8}$$

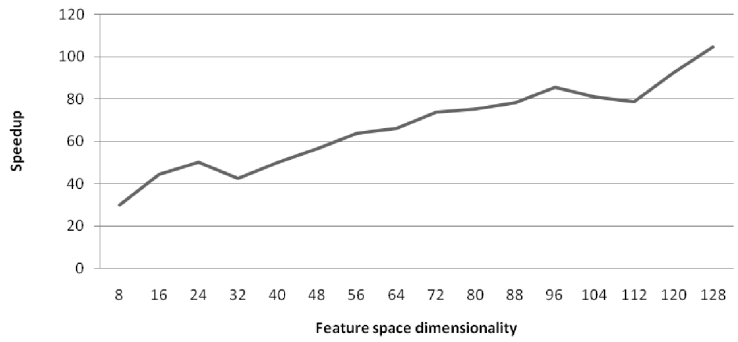


Figure 4.18: Speedup measured as CPU/GPU ratio of fuzzy c-means clustering using 4096 data points, 64 cluster and varying the feature space dimensionality.

Table 4.2: Time in seconds for various profiles on the Nvidia Tesla C2070.

Clustering profile	**Time in seconds**
C=4, DP=4096, F=4	0,005
C=4, DP=4096, F=128	0,009
C=64, DP=4096, F=4	0,021
C=64, DP=8192, F=4	0,029
C=16, DP=40960, F=32	0,124
C=4, DP=409600, F=8	0,258

In addition, a very useful version of this algorithm was designed by the creator of the game Quake 3 and then motivated in Lomont (2003). It uses as initial guess the *magic* value $0x5F3759DF$, which has proven to give as outcome, after a single iteration, an average error around 10^{-6}, and a maximum error around 10^{-3}. In addition, how can be seen from Figure 4.19 it is around two times faster than the standard $sqrt()$ function.

The used square root approximation exploits the IEEE 754 log-style floating point format, in particular the code used is reported in the following snippet:

Listing 4.1: Listing for the code used for square root function optimization.

```
float SquareRootFloat(float number) {
 long i;
 float x,y;
 const float f=1.5F;
```

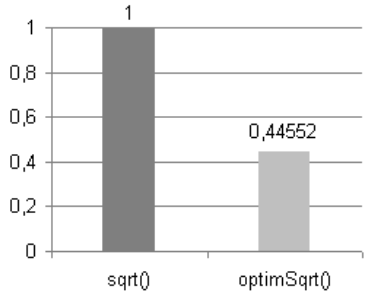


Figure 4.19: Comparison of average execution time between standard and optimized square root functions applied to 1000 samples. Values are normalized w.r.t. reference *sqrt*() implementation.

```
 x=number*0.5F;
 y=number;
 i=*(long*)&y;          // get floating value bits
 i=0x5f3759df-(i>>1);// initial guess
 y=*(float*)&i;         // re-conversion to float
 y=y*(f-(x*y*y));       // Newton step
 return number*y;
}
```

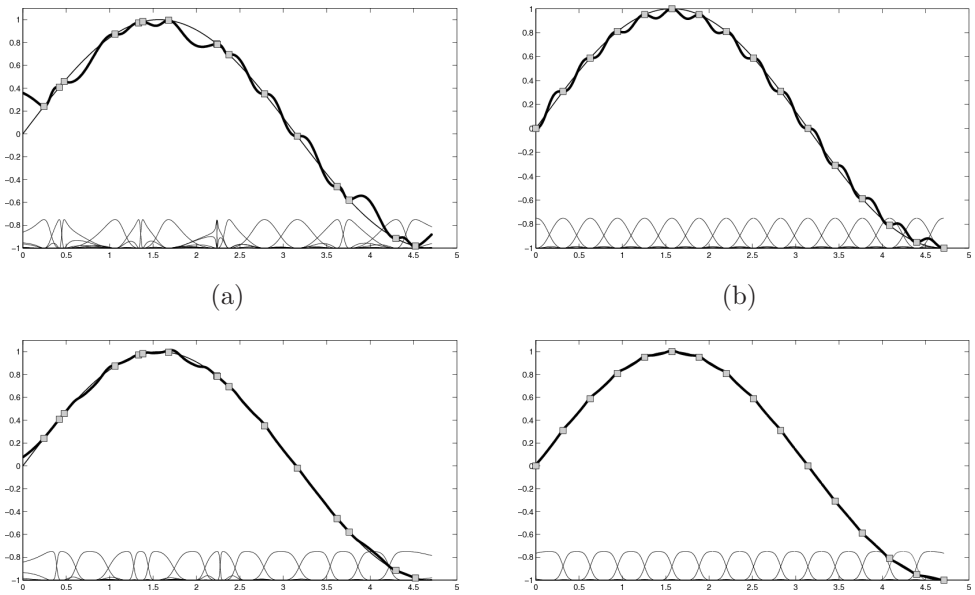
Chapter 5

Results and discussion

The framework and all of its applications have been extensively tested in order to evaluate their performance both quantitatively and qualitatively. First experiments were conducted on the theoric kernel regression framework, to determine its precision and applicability. Framework applications are then tested, evaluated and with toy examples and both simulated and real datasets. Successively follow some tests on the interpolation scheme proposed, evaluating the suppression of MI local minima resulting from its application. Lastly, a qualitative evaluation from an expert radiologist is given too, in order to provide a feedback directly from the final system user.

5.1 Tests on Fuzzy Kernel Regression

Before doing any test with images, we needed to validate the Fuzzy Kernel Regression framework. In order to do this, we evaluated simple mono-dimensional function regression performance and evaluated the error induced by the three strategies used: direct interpolation from random function samples representing the landmark displacements for *simple landmark based registration* in Figure 5.1a (for comparison purposes we evaluated the error even for equally spaced samples in Figure 5.1b), regression used for smoothing piece-wise linear interpolation from random function samples for *improved landmarks based registration* in Figure 5.1c, and regression used for smoothing piece-wise linear interpolation from equally spaced function samples for *automatic area-based registration* in Figure



(a) (b)

(c) (d)

Figure 5.1: 1*d* Function regression results. *simple landmark based* (a, b) and *improved landmarks based* (c, d). Results are shown using random samples (first column), equally spaced samples (second column). On the bottom of each diagram fuzzy kernels shapes are plotted.

5.1d. In each of the pictures is shown the recovered function from 16 known samples along their relative fuzzy membership kernels (on the bottom). As it can be seen, regression operated directly from samples values leads to some oscillations, while smoothing piecewise linear interpolation leads to a much better estimation of the samples underlying function. These results were quantitatively measured computing the Root Mean Square error of the regression function w.r.t. the exact function. Results are reported in Table 5.1 for several type of limited codomain functions.

As it is evident from the results, the improved method outperforms simple one. In addition, the lowest the variance of the distances between the known function samples, the lowest is the resulting regression error, getting very low when it approaches 0. Errors in simple method are mainly due to the fluctuations occurring as getting further from the samples. The biggest the distance between two samples, the largest are the fluctuations. In improved method this effect

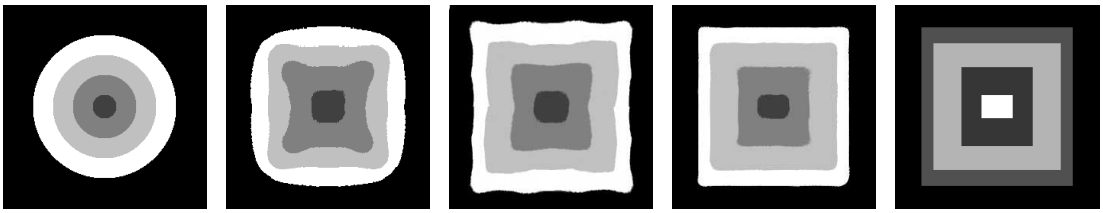
Table 5.1: Root Mean Square errors for Simple and Improved Fuzzy kernel regression, results are given both for randomly spaced ($\sigma_{dist} = 1.9$) and equally spaced

Regression errors, RMS (%)				
	Simple		Improved	
Function	Random	Equispaced	Random	Equispaced
$sin(x)$	7.70	3.84	1.60	0.61
$sinc(x)$	5.18	3.50	1.78	1.15
$sigmoid(x)$	1.76	0.92	0.19	0.06
$gauss(x)$	4.38	2.83	1.20	0.56

is removed because the values are constrained by linear interpolation. This is a theoretic basis which confirms how these methods can be used for recovering registration functions.

5.2 Quantitative results

In order to validate the performance of the registration framework, several tests were conducted using the three applications proposed with 2d datasets. In addition 3d tests for the area-based automatic method were conducted too. The datasets used for the experiments are both synthethic and real. For synthethic data we used the Brainweb generator Cocosco *et al.* (1997); Collins *et al.* (1998); Kwan *et al.* (1996, 1999), while real datasets were obtained from the Oasis database Marcus *et al.* (2007) and scans provided by "Ospedale Civico di Sciacca". The registrations were done using the proposed distance based interpolation method, however its validation is done separately after the registration tests. The type of the images used are CT and PD-, T1- and T2-weighted MRI. For each experiment, results of *Simple landmark-based approach (SLB)*, *Improved landmark-based approach (ILB)* and *Automatic area-based approach (AAB)* are compared.



(a) input (b) SLB result (c) ILB result (d) AAB result (e) target

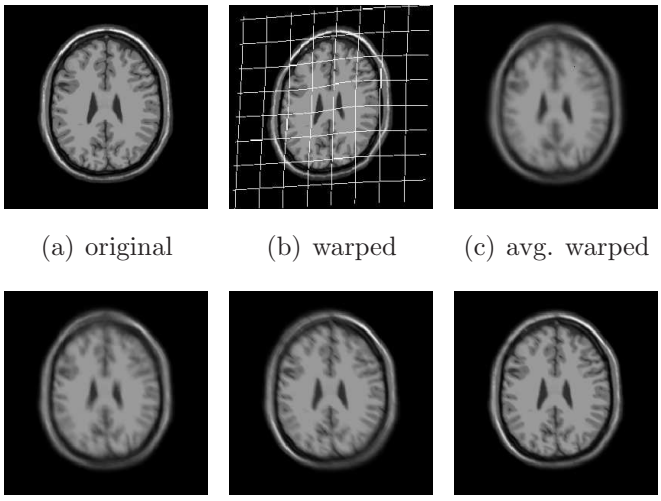
Figure 5.2: Theoretic multimodal example results. The input circles (a) are registered to the target squares (e). Results for simple landmarks based (b), improved landmarks based (c) and area based (d) methods are shown.

5.2.1 Theoretical multimodal example

Before starting the actual experiemnts, some registration tests were operated on toy example. Using as test images two multi-modal patterns (concentric circles as input, Figure 5.2a, and squares as target, Figure 5.2e), we computed the required aligning transformation using the three registration methods. For the landmarks based method each circle was marked with eight equally spaced points. Using SLB (Figure 5.2b), the corners are not well aligned due to the fluctuations induced by raw fuzzy kernel regression. With ILB (Figure 5.2b), corners are better aligned due to the regularization effect provided by affine transformations composition. Lastly, with AAB many iterated affine transformations grant a more correct corner and edge alignment.

5.2.2 Synthetic Multimodal registration

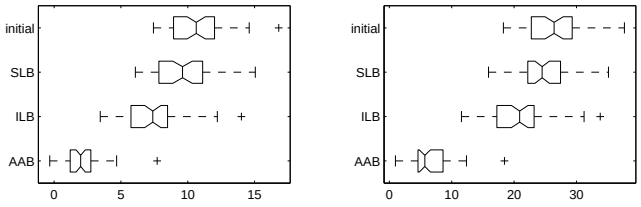
In this second experiment, we intended to evaluate the performance of the registration schemes on synthetic data. To generate the pairs of images, we start from an unmodified image obtained from the Brainweb database. We then produce some artificial random deformations with a maximum amplitude of 20 pixels. This is done by means of Thin-plate Spline surfaces. Such deformations are applied to the target image which is succesively registered back. The error is evaluated computing the average intensity differences (AID) and the root mean square (RMS) of the local registration error in each voxel. The results for this experiment is reported in Figure 5.3 and Figure 5.4. Figure 5.3 shows the original image, an ex-



(a) original (b) warped (c) avg. warped

(d) avg. SLB (e) avg. ILB (f) avg. AAB

Figure 5.3: Average of the results obtained over 200 registrations.



(a) AID (b) RMS

Figure 5.4: Boxplot of the registration results over 200 image pairs. The graphs show the distribution of the average intensity difference (AID) and the root mean square (RMS) of the registration error. Registrations were performed using the application of the proposed framework in its three versions: simple landmark based (SLB), improved landmark based (ILB) and automatic area-based (AAB).

ample of a deformed image, the average of the transformed brain images and the average of the registered brain images using the three proposed methods. Sharper images indicate that the registrations are, in average, more accurate. In addition, in Figure 5.4 are reported boxplot diagrams for AID and RMS indexes. The same results, along with relative computation times, are summarized in Table 5.2.

Table 5.2: Results for the experiments shown in Figure 5.3

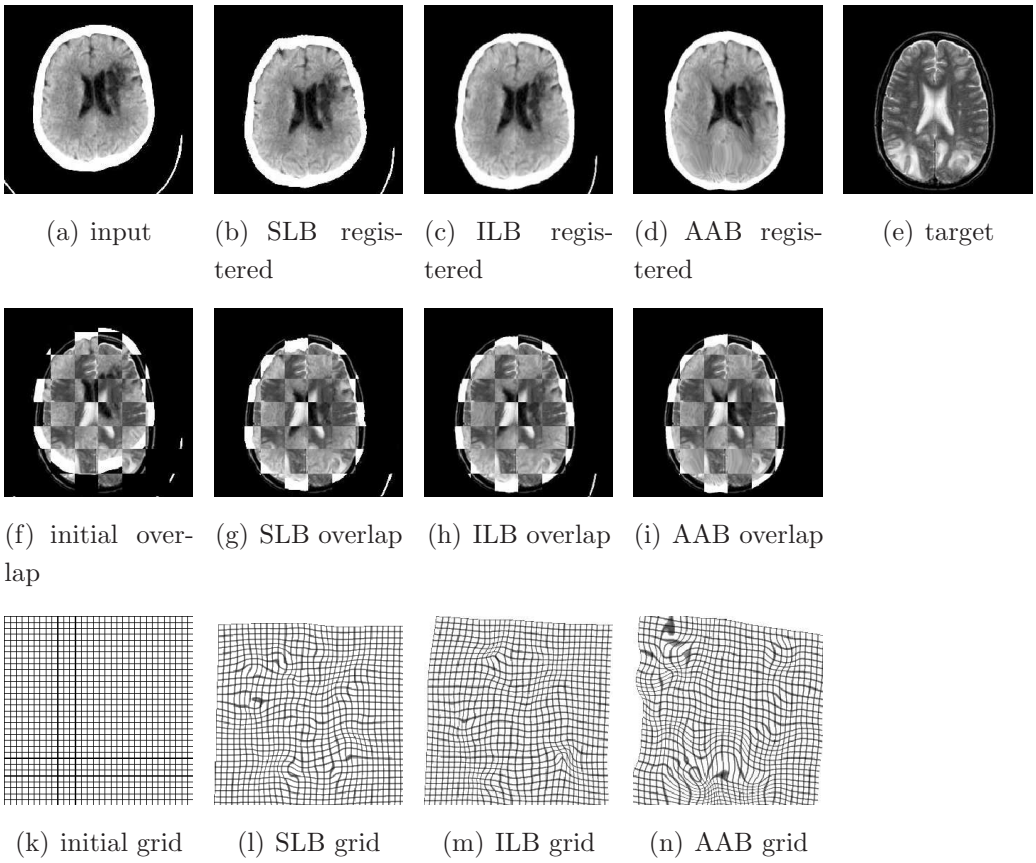
	AID	RMS	time (s)
SLB	9.89 ± 2.42	25.99 ± 4.28	11.70 ± 0.09
ILB	7.63 ± 2.34	21.26 ± 5.12	13.58 ± 0.05
AAB	2.18 ± 1.41	8.43 ± 3.35	165.39 ± 12.11
initial	11.15 ± 2.64	26.89 ± 5.30	

5.2.3 Real Multimodal registration

Last registration experiment regards real multi-modal images. Images acquired with different technology equipment were involved in the registration process. To validate the robustness of the system inter-patient images with extremely different anatomies were used. Test cases patients present eventually also pathologies or diseases which vary drastically the intensity level distribution in the image. An example of such registrations is shown in Figure 5.5 for the three applications of the framework. Note the different head shape and the stain (Figure 5.5a). SLB registration (Figure 5.5b) and ILB (Figure 5.5c) succeed in registering anatomical parts of the brain (if landmark points are well-determined) but does not fully recover the shape of the head. AAB (Figure 5.5), allowing free-form like transformations, deforms the structure to achieve succesful alignment of the whole anatomy.

5.2.4 Sequential registration

Next experiment regards images of the same patient acquired in different time presenting anatomic differences (for example due to resections). Tests were operated on 30 CT case studies. An example, using AAB registration is shown in Figure 5.6. As can be seen in this case study, the pose unalignment and the large differences in the ventricles are diminished granting a complete overlap of the target (Figure 5.6b) and registered (Figure 5.6c) images using a checkerborad visualization (Figure 5.6d).



(a) input (b) SLB registered (c) ILB registered (d) AAB registered (e) target

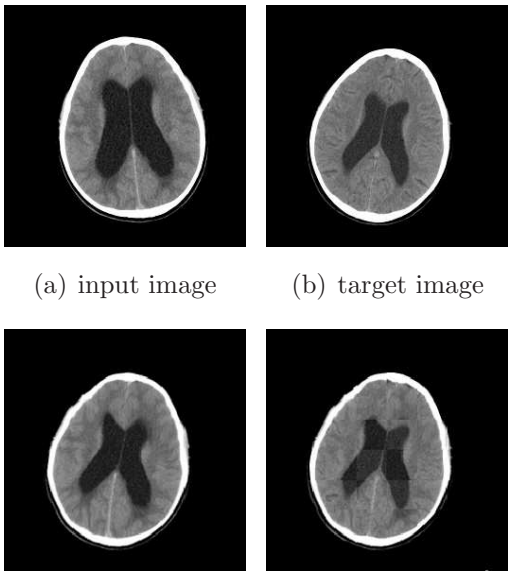
(f) initial overlap (g) SLB overlap (h) ILB overlap (i) AAB overlap

(k) initial grid (l) SLB grid (m) ILB grid (n) AAB grid

Figure 5.5: CT-T2 registration example test. In the first row, input image (a) is registered onto target image (e). Results for simple landmarks based (b), improved landmarks based (c) and area based (d) methods are shown. In the second row, checkerboard visualization for alignment is depicted: in (f) for input and target image, in (g), (h) and (i) for simple, improved landmarks based, and area based approaches respectively. Last row (k-l) represents the deformation grid for the recovered transformations.

5.2.5 Tests on Interpolation

Existing literature studies, such as Tsao (2003) and Ji *et al.* (2003), generally evaluate algorithms using rigid transformations. Due to the large proliferation of non-rigd and elastic registration techniques, it is important to assess interpolation schemes performance using elastic deformations instead. For this reason, experimental tests on the proposed interpolation were performed using several



(a) input image (b) target image

(c) registered image (d) overlap

Figure 5.6: Registration results for different time inter-patient scans. Floating image (a), reference image (b), registration result (c), checkerboard overlap (d).

type of parametric transformation, which, although are not likely to exist globally in reality, they can occur locally in real cases. For each transformation the trend of the MI measure has been evaluated both qualitatively and quantitatively. From a qualitative perspective is sufficient to visually inspect the shape of the function and evaluate the amount of local maxima. For a quantitative assessment, a smoothness measure was used as done in Tsao (2003). This measure is computed as the inverse of the root mean square (RMS) error of the difference between the MI functional and its smoothed version (Figure 5.7). The proposed interpolator (DI) and its jittered version (DIJIT) were evaluated and compared against nearest neighbor (NN), jittered nearest neighbor (NNJIT), linear interpolation (LI) and partial volume interpolation (PVI), which are the current best literature tradeoffs between quality and speed. The transformation used for the experiments are polynomial, pinch/spherize and twirl. In Figure 5.8 are shown examples of such transformations. As previously remarked, MI estimate is based on joint histogram computation. Such histogram was evaluated for a different number of bins (N = 8, 16, 32, 64, 128, 192, 256) for each interpolation scheme.

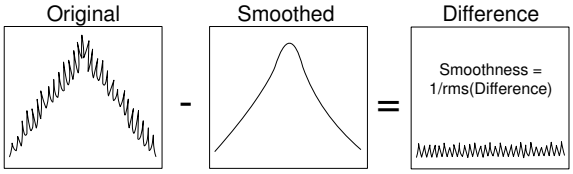


Figure 5.7: Quantitative measure of interpolation artifacts. Left: original MI curve. Center: smoothed version. Right: difference between original and smoothed curves. The result is mainly the amount of artifacts generated in the interpolation process. Smoothness is estimated as the inverse of rms of the difference curve.

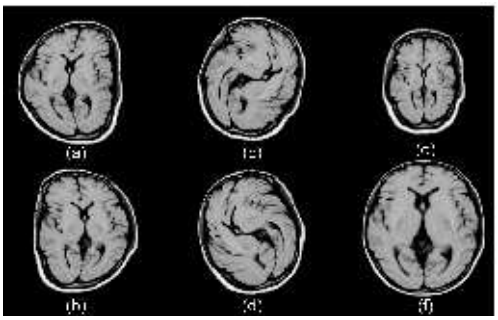


Figure 5.8: Type of deformations used for performance evaluation: polynomial transformation (a-b), twirl transformation (c-d), pinch/spherize transformation (e-f).

The independent parameter of the transformations was adjusted in a convenient range (for example twirl transformation varies the rotation angle in the range $\pm 40°$).
Resulting registration curves plots are reported in Figure 5.9. As can be seen NN keeps quite smooth but eventually exhibits some step-like artifacts when approaching the correct alignment. However their presence in elastic deformations is smaller than in rigid transformations due to the non-regular image grid deformation. As known this problem is mitigated with jittered NN. Artifacts in the MI curves are reduced, but some random fluctuations are added. Both LI and PVI can suffer from local maxima/minima presence problems. For what concerns DI the curves keep smooth and the local maxima/minima are less pronounced and their presence results even more reduced using jittered sampling. Another

remarkably aspect regards the numbers of bin used for joint histogram estimation. Using more bins does not necessarily correspond to a better MI estimate. From Figure 5.9 it can be seen that the optimal number of intensity bins is around 32-64. Using more bins does not improve the estimate since the number of samples populating each bin is too low to give an accurate measure. Such effect was already reported in Ji *et al.* (2003), where is suggested to use intensity clustering, which is actually equivalent to use less intensity bins or to decrease the quantization levels. Another advantage of intensity clustering is to reduce the effect of noise.

Such effect is visible and quantitatively appreciable analyzing the smoothness measure plots. Results for the six interpolation methods are depicted and compared in Figure 5.10. The plots show the trend of the smoothness assessment for each of the three deformations applied, as a function of the number of intensity bins. For polynomial and pinch/spherize transformations it results that even from this perspective, a number of bins around 64 represents the optimal choice, except for DIJIT and PVI, for which the maximum value is achieved using around 128 bins. Note that notwithstanding using 8 bins provides an higher smoothness value, the MI estimate is very poor since all the samples are flattened due to an excessive loss of information. For what concerns twirl transformation, the trends are quite different, this is due to the high deformation introduced and the consequent increase of the interpolation errors. In this case, decreasing the numbers of bins will not help since the resampling error induced is too high, in fact the smoothness measure values obtained are substantially lower than in the other two cases.

It is remarkable that in all the three cases, distance interpolation and its jittered version keep more performant than the others or at least comparable.

Last considerations are about timing performances. Average computational times were measured on a AMD Phenom Quad-core, 2.30Ghz equipped with 4gb of RAM. Results are reported in Figure 5.11, values are normalized w.r.t. NN interpolation. Except from NN and NNJIT which are very quick to compute, other interpolation schemes have comparable timings. DI and DIJIT, thanks to the optimized function used, keep their computation time quite low, allowing to use the methods as a valid alternative in practical aplication even from an efficiency point of view.

Figure 5.9: Registration curves. In each row is represented a different deformation, from top to bottom: polynomial, pinch/spherize, twirl. In each column is represented an interpolation scheme, from left to right: Nearest Neighbor, Jittered Nearest Neighbor, Linear, Partial Volume, Distance, Jittered Distance. In each plot are represented the registration curves varying the number of histogram bins. The used values are 8, 16, 32, 64, 128, 192, 256.

Figure 5.10: Smoothness values for different interpolators as a function of the intensity bins used for joint histogram. Values are reported in semi natural-log scale. Each row corresponds to a different applied deformation, from top to bottom, polynomial, pinch/spherize, twirl. In each plot, the curves represent the smoothness of the relative registration curve using different interpolators.

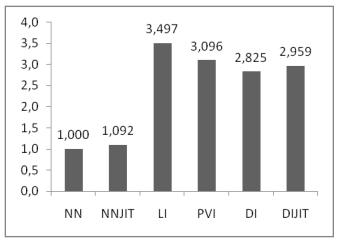


Figure 5.11: Timing performance for the considered interpolation schemes. Values are normalized w.r.t. NN interpolation.

5.3 Qualitative results: expert validation

In order to add value to the framework and the proposed systems, we submitted our results to the evaluation from expert radiologists. Such activity was belived very important for estabilishing whether such results are satisfying not just from a numerical or visual perspective, but also from the usage of the system on real diagnosis tasks. The evaluation was based on the following level of assessment:
- *Global alignment rating*: The overall evaluation of the registration procedure, in terms of shapes and contours matching.
- *Availability of points of interest*: The chance to found the points of interest in the very same positions of the images. For each test case a list of features are given and their alignment after registration is rated.
- *Morphological structures coherency*: The coherence of the morphology after the registration of the image is rated to report any structure deformation or anomaly.
For the evaluation 30 test cases were used, each one consists in a pair of MRI datasets (T1-, T2- or PD-weighted). In addition to the existing unalignment a further random roto-translation is injected to stress the method. Results for these tests are reported in Table 5.3. Each row reports the test number, the image type, the points of interest considered and the rating for the three levels of assessment. Automatic area-based registration was used for this test. Some of the low-rated test cases came out to be outliers, since after the artificial roto-translation some peripheral information lied out of the cropping bounds. Consequently, some of the structures got lost and cannot be realigned, compromising (even if just locally) the whole registration task.

Table 5.3: Expert evaluation for the registration procedures

Test	Image type	Points of interest	Global align.	Pts.of interest	Morph.coherency
1	Axial T1 vs T1	Lateral v., corpus c., frontal g.	5	5	5
2	Axial T1 vs T1	Lateral v., 3rd v., basal ganglia, thalamus	5	5	5
3	Axial T1 vs T1	4th v., bulb, cerebellar hemisp., maxillary sinus	5	5	5
4	Coronal T1 vs T1	Lateral v., basal ganglia, Sylvian fissure, opt. chiasm and tract	5	5	5
5	Sagittal T1 vs T1	Temporal g., cerebellum	5	5	5
6	Axial T1 vs T1	4th v., bulb, cerebellar hemisp.	5	5	5
7	Axial T2 vs T1	Lateral v., corpus callosum, frontal g.	5	4	4
8	Axial T2 vs PD	Acqueduct, midbrain, basal cistern, cerebellar hemisp.	4	5	4
9	Axial T2 vs T1	Lateral v., basal ganglia, thalami	5	5	5
10	Axial T2 vs PD	4th v., bulb, cerebellar hemisp., maxillary sinus	5	5	5
11	Axial T1 vs T1	Fronto-parietal g.	5	5	5
12	Axial T2 vs T1	Basis pontis, 4th v., acustic nerve, cerebellum, ocular bulbs	5	5	5
13	Axial T2 vs PD	Basis pontis, 4th v., acustic nerve, cerebellum, ocular bulbs	5	5	5
14	Coronal T1 vs T1	Frontal g., orbital fat	4	5	3
15	Coronal T1 vs T2	Frontal g., orbital fat	4	5	4
16	Coronal PD vs T2	Parieto-occipital g., cerebellum, sup. sagittal sinus	4	5	3
17	Sagittal T1 vs T2	Temporal lobes, cerebellum	5	5	4
18	Sagittal T1 vs T1	Temporal lobes, cerebellum	4	5	4
19	Sagittal T2 vs T2	Occipital horn, cerebral g., cerebellum	3	5	3
20	Sagittal PD vs T1	Occipital horn, cerebral g., cerebellum	5	5	4
21	Axial T1 vs T1	Midbrain, Cerebellar vermis, temporal g., ocular bulbs, opt.nerves	3	5	4
22	Sagittal T1 vs T1	Fronto-temporal g., cerebellum, maxillary sinus	5	5	5
23	Sagittal T1 vs T1	Corpus callosum, fornix, brainstem, cerebellum, pituitary	5	5	4
24	Sagittal T1 vs T1	Corpus callosum, fornix, brainstem, cerebellum	5	5	5
25	Axial T1 vs T1	4th v., temporal lobes, cerebellum	5	5	5
26	Axial T1 vs T1	Brainstem, temporal lobes, cerebellum	4	5	4
27	Coronal T1 vs T1	cerebellum, occipital g.	3	5	4
28	Coronal T1 vs T1	cerebellum, occipital g.	5	5	5
29	Sagittal T1 vs T1	Corpus callosum, brainstem, cerebellum	4	5	5
30	Sagittal T1 vs T1	Fronto-temporal g., cerebellum, maxillary sinus	5	5	4
Mean	-	-	4.57	4.97	4.40

Chapter 6

Beyond Image Registration: Content aware image resizing

While working on image registration, it turned out to be clear that the same approach can be applied to other deformable image transformations. In this direction, the first problem approached is "content aware image resizing", i.e. the problem of resizing an image, changing its aspect ratio, without deforming relevant content.

6.1 Content aware image resizing

With the advent and proliferation of display devices which come with different aspect ratio and resolutions, automatic resizing is becoming an important issue. Applying image cropping is not sufficient due to information loss. Arbitrary resizing, which produces deformations, is not suitable as well. Any approach that applies homogeneous transformations to each image region, will spread distortion equally. An alogorithm for content-aware image resizing should preserve relevant regions of the images, introducing distortions just in the regions where no important content elements are present.

Recently, some techniques were proposed. Seam carving Avidan & Shamir (2007); Rubinstein *et al.* (2008) removes or inserts discrete *1D* seams passing across the less important regions in the image. Warping methods Gal *et al.* (2006); Wolf

et al. (2007); Yu-Shuen Wang & Lee (2008) introduce mesh grid that get warped according to a functional which keeps unchanged important regions as much as possible. Combination of several operators try to apply them in an adaptive fashion Rubinstein *et al.* (2009). The first approach has intrinsic limitations due to its discrete nature, which limit its effectiveness to cases where the content spreads in a region smaller than the new image size, the second one becomes as more expensive to compute as the grid size and/or the image resolution increase, thus resulting inefficient for large or detailed images.

We present an efficient method which, using a simpler approach, can achieve results superior or comparable with literature methods. It consists in determining shift values for each column/row in the image such that distances between relevant columns/rows are left unchanged. In this way, distortions are reduced to the minimum, while content can get scaled in order to fit arbitrary image sizes. Relevant columns/rows are determined using a measure that can be derived using several relevance maps, such as visual saliency map Itti *et al.* (1998), corner detectors Harris & Stephens (1988), eye-gaze measurement Santella *et al.* (2006), etc. The process requires the solution of a simple linear system with a limited number of variables, equal to the number of columns/rows. This can be efficiently solved allowing real-time usage.
In addition the method can been improved by adding both automatic or interactive cues to the system solution: a face detector Viola & Jones (2001) can help in preserving faces in the images, other geometric constraints can be given by the user to explicitly preserve structures.

6.2 Related work

Normal resizing operators used by image processing applications generally work by resizing images to a target size by using an homogeneous a shrinking or enlarging operator. In literature several works have been propsed for image retargeting. First attempts were done using automatic cropping Suh *et al.* (2003) where the most important region of the image is determined using a saliency map or a face detector before cropping it to show only the most salient region in the image. Another solution, proposed in Liu & Gleicher (2006), is to compute an optimal

path through the most salient region of the image and to display them serially on low resolution mobile devices. Such techniques for changing the image size, always rely on standard resizing and cropping methods. More recent approaches use adaptive image resizing instead. The idea is to preserve important image features by applying a non-linear content driven resize operator. Most remarkable works were done using seam carving, Avidan & Shamir (2007); Rubinstein *et al.* (2008) where 1D seams are removed/added to reduce/increase the image size. Such seams are chosen from low energy regions of the image. The result is very impressive. However, due to the discrete nature of the method, notches in the objects may appear. In addition when no more discardable information exists, important details get removed and severe distortion appears. Warping methods Gal *et al.* (2006); Wolf *et al.* (2007) overcome this limitation by squeezing or stretching homogeneous regions, minimizing the distortion in relevant regions. In Yu-Shuen Wang & Lee (2008) regions are scaled by different factors in order to preserve aspect ratio too. Multi-operator approach Rubinstein *et al.* (2009), uses a combination of seam carving, scaling and cropping. Seam carving is very efficient but limited in its use, warping methods are more effective but computationally expensive, limiting their use for real-time applications with high resolution images or embedded devices with low power profiles.
Our work aims to design a non-linear resizing operator which can deal effectively with content preservation while requiring efficient and fast computation.

6.3 Proposed solution

In our model a set of m by n grid points $L = [l_{0,0}, l_{0,1}, ..., l_{m-1,n-1}]$ is superposed to the image, where $l_{i,j}$ are the initial points positions and m and n are the initial sizes. To resize the image to the new dimension $m'n'$ we look for the new set $L' = \left[l'_{0,0}, l'_{0,1}, ..., l'_{m-1,n-1}\right]$ where distances between two neighboring points are preserved (6.1) in order not to introduce distortions.

$$\begin{aligned} (l_{i,j} - l_{i-1,j}) &= \left(l'_{i,j} - l'_{i-1,j}\right), \\ (l_{i,j} - l_{i,j-1}) &= \left(l'_{i,j} - l'_{i,j-1}\right). \end{aligned} \tag{6.1}$$

Obviously, due to image resizing, some distances should be necessarily changed and some deformation must be introduced. The model is built in order to spread

the required deformations non-uniformly across the whole image, according to lines significance.

6.3.1 Region significance

Literature image retargeting methods rely on various significance measures, Avidan & Shamir (2007) and Wolf *et al.* (2007) consider large gradient magnitudes regions as significant, Rubinstein *et al.* (2008) uses the accumulation of the discontinuities of the neighborhood if a given pixel is removed. In Yu-Shuen Wang & Lee (2008) a combination of gradient information and visual saliency maps are used Itti *et al.* (1998). Such measure, that was used in this work, is defined as $W = W_\alpha \times W_\beta$, where $W_\alpha = \sqrt{\left(\left(\frac{\partial}{\partial x}\right)^2 + \left(\frac{\partial}{\partial y}\right)^2\right)}$ is the 2-norm of the gradient and W_β is the saliency map. This map is computed by applying various filters to extract properties such as color, intensity and orientation, and then searching neighboring regions exhibiting differences in these properties on multiple scales.
The measure can be obtained at different resoultion scales. In our model we work on a tessellation of sub-regions, so saliency is computed for variable size areas, one value for each of them.
Note that, even though we adopted this measure, any different one can be used without loss of generality.

6.3.2 Reduced linear system model

Given the new image size, we compute the new points positions, such that distances between points containing prominent objects are left unchanged, while distortions are applied to low-importance lines. The new disposition should be subject to boundary constraints. We formulate the distortion energy for each point $l_{i,j}$ measuring how much varies the distance from the neighboring point $l_{i-1,j}$ and $l_{i,j-1}$. The total energy function can then be defined as:

$$D_u = \sum_{i=1}^{n-1} \sum_{j=1}^{m-1} \left[\left(l'_{i,j} - l_{i-1,j}\right)' + \left(l'_{i,j} - l'_{i,j-1}\right) \right]. \tag{6.2}$$

This formulation alone will produce homogeneous resizing, then, in order to achieve content-aware resizing each term should be multiplied by its weighting

factor $w_{i,j}$ defined by visual saliency. (6.2) then becomes:

$$D_u = \sum_{i=1}^{n-1}\sum_{j=1}^{m-1}\left[\left(l'_{i,j} - l_{i-1,j}\right)' + \left(l'_{i,j} - l'_{i,j-1}\right)\right] w(i,j). \tag{6.3}$$

Expanding (6.3) and factoring its results produces:

$$D_u = -\sum_{i=1}^{n-1}\sum_{j=1}^{m-1}\left[\left(w_{i-1,j} - w_{i,j}\right) l'_{i-1,j} + \left(w_{i,j-1} - w_{i,j}\right) l'_{i,j-1}\right]. \tag{6.4}$$

Since weighting factors $w(i,j)$ are known, (6.4) represents the objective function of the linear model. The model needs to be constrained for several reasons: (note that similar considerations hold both for shrinking and enlargement).

1) The procedure, while attaining the minimization would increase the distance in high saliency regions, while it should be left unmodified (i.e. equal to the grid element size s).

The inequalities expressing this constraint is the following:

$$\begin{matrix} l'_{i,j} - l'_{i-1,j} \leq s \\ l'_{i,j} - l'_{i,j-1} \leq s \end{matrix}, \quad \forall i,j \in [1, m-1; 1, n-1]\,. \tag{6.5}$$

2) Low relevance lines distances can get compressed to 0, and this can produce unwanted discontinuities and artifacts, so a minimum/maximum distance should be assured:

$$\begin{matrix} l'_{i,j} - l'_{i-1,j} \geq k \\ l'_{i,j} - l'_{i,j-1} \geq k \end{matrix}, \quad k \in [0,1]\,, \quad \forall i,j \in [1, m-1; 1, n-1]\,. \tag{6.6}$$

3) Boundary conditions must be respected to enclose the image into the new size:

$$\begin{aligned} l'_{0,j} &= 0, \quad \forall j \in (0, n-1)\,; \\ l'_{m-1,j} &= m', \quad \forall j \in (0, n-1)\,; \\ l'_{i,0} &= 0, \quad \forall i \in (0, m-1)\,; \\ l'_{i,n-1} &= n', \quad \forall i \in (0, m-1)\,. \end{aligned} \tag{6.7}$$

The optimization is performed using a primal-dual interior-point method Mehrotra (1992) which converges iteratively to the optimal solution. Experiments shown that the displacements of neighboring columns should be similar, then, to avoid sharp differences in columns displacements, the values provided by the optimization are low-pass filtered to achieve a smooth displacment distribution and a better visual result.

6.3.3 Fuzzy Kernel Regression application

After the optimization of (6.4), the new grid points positions L' are recovered and their disposition on the new image is determined. On this basis, the underlying image pixels have to be interpolated to move them in the resized image (see Figure 6.1). This situation is the same as the image registration problem, where points correspondences are known. Determining from (4.2) the affine transformation which maps each grid patch, fuzzy Kernel Regression framework can be applied, using (4.3), to recover the mapping function and obtain a smooth global mapping function for the whole image.

Figure 6.1: Grid deformation after minimizing the content distortion energy.

6.4 Results

The described method was implemented on a PC with Quad CPU 2.30 GHz. Since the method relies on the solution of a simple linear system, the computation is very efficient. In Figure 6.2 is shown the timing function diagram varying the size of images with aspect ratio 4:3. This result points out that the method can be used for real-time usage even for large image resizing purposes, especially if ported to an executable application. In addition, large image can be also fastly resized using downsampling ratios without substantially altering the final result.

Comparison. In order to evaluate the results of our system, we compared it

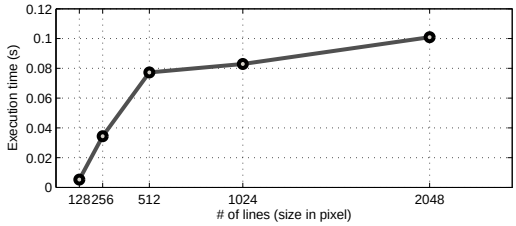


Figure 6.2: Time elapsed for resizing an image. Values are function of the number of lines involved in the process.

(a) Original image

(b) 50% width image

(c) 150% width image

Figure 6.3: Examples of the resizing algorithm. Original image (a), shrinked (b) and enlarged (c) images .

with other literature retargeting systems: Multi-operator Rubinstein *et al.* (2009), Non-homogeneous warping Wolf *et al.* (2007), seam carving Rubinstein *et al.* (2008) and scale-and-stretch Yu-Shuen Wang & Lee (2008) using the datasets and measures provided by *RetargetMe* framework Rubinstein *et al.* (2010). Examples of such comparisons are reported in Figure 6.4, where can be seen how less deformation are present in the shown images (most evident case is child's head in third row of Figure 6.4). Beside visual inspection, which is intrinsically subjective and not easily evaluable, an objective analysis was conducted to as-

Figure 6.4: Comparisons with reference algorithms. From left to right: original image (a), proposed (b), Multi-operator (c), Non-homogeneous warping (d), seam carving (e) and scale and stretch (f).

sess the quality of the results. Two measures were used for this purpose: Earth Mover's Distance (EMD) Pele & Werman (2009) and SIFTflow Liu *et al.* (2008), two commonly used similarity metrics which does not require the two dataset to be the same size, a binding property for the case of image retargeting. Results, reported in Figure 6.5 and summarized in Table 6.1, show that the images produced with the proposed method provide in average measures comparable to literature methods, or even better. Considering these results, the main strenght of the proposed method is that just linear algebra operations are needed.

Table 6.1: EMD and SIFTflows measures for images of *RetargetMe* framework

Measure	EMD	SIFTflow
Proposed	$8.13 \pm 3.36 \cdot 10^3$	$4.15 \pm 2.12 \cdot 10^5$
Multi-operator	$8.30 \pm 3.58 \cdot 10^3$	$3.94 \pm 1.99 \cdot 10^5$
Non-homogeneous	$8.68 \pm 3.73 \cdot 10^3$	$4.12 \pm 2.15 \cdot 10^5$
Seam carving	$8.69 \pm 3.60 \cdot 10^3$	$4.09 \pm 2.38 \cdot 10^5$
Scale and stretch	$8.95 \pm 3.82 \cdot 10^3$	$5.37 \pm 2.69 \cdot 10^5$

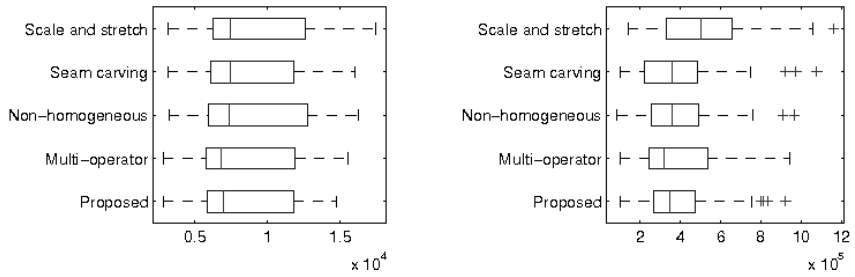


Figure 6.5: Comparisons: boxplot diagrams over 37 images, values from reference algorithms are provided by *RetargetMe* framework. Last row shows values from springs network retargeting.

6.5 Web implementation

Generally, web designers have to deal with the potential access of their pages from very different devices, spanning from desktop PCs, laptop, netbook, smartphones, other mobile devices, etc. For this reasons image size compliance should be provided for almost any display device. Traditionally there exist two possible approaches:
- *Conservative*: the webpage layout is developed in order to be compatible with the smallest device considered, however this results in a bad use of large displays.
- *Multiple views*: the developer realizes several page views, displaying to the user the most suitable one according to its display device. Even though this choice produces better results, it is very time-consuming in the development and maintenance phases.
A third innovative approach could be to use liquid layouts which dynamically resize both textual content and images. However, current web browsers are able to perform natively just simple homogeneous resizing. Obviously this kind of transformation is not suited because, being applied equally in each image region, will spread distortions uniformly across the whole image. In the same way, image cropping technqiues are not sufficient since they may result in a severe loss of information. For this task the proposed algorithm was implemented as a web-service to dinamically adjust image size according to the available visualization device

Natively, the HTML <*IMG*> tag does not allow manipulation of an image at pixel level. Hence, JavaScript has to be used to push retargeting functionality into the page. A simple solution can be to replace all of the <*IMG*> tags in the page with Flash objects, or with the HTML5 canvas elements, thus allowing pixel level manipulation. However, both solutions are not applicable since are really time consuming and can freeze the browser for a long time. For these reasons, all of the computation should be done at the server side, leaving to the client just the task of loading the resized images from a service.
The designed application is composed by two parts. A server side, implemented as a web service, which encapsulates the image resizing procedure, and a client side, which, trasparently to the developer, performs the request for the resized image.

6.5.1 Server side

The server side is realized by a Java servlet which receives a GET request from the client containing the following input:
- *Image URL*: the URL of the requested image.
- *Width*: The requested width (in pixels) for the resizing task.
a typical servlet request will look like this:

```
http://www.hostname.com?http://path.to/image.jpg&750
```

the servlet, given these parameters, performs the content-based image retargeting procedure and returns the resized image bytes in the form of an HTTP response.

6.5.2 Client side

A webpage can call directly the servlet using the syntax described in 6.5.1. However, this requires the developer to know such syntax and does not allow dynamic resizing of the images (i.e. image size needs to be chosen at design time).
For this reasons a different solution is required for allowing both the following:
- Automatic width selection w.r.t to the display device or window size.
- Dynamic resizing triggered by window resizing event.

A seamless way to code images which needs to be resized would be the following:

```
<body>
...
  <img class="retarget" src="path_to_img.jpg" width="20%" />
...
</body>
```

Natively, specifying the width in percentage, resizing will be performed homogeneously, thus leading to severe distortions. In order to achieve content-aware image resizing, each time that the window *onresize* and the *onload* events are fired, images labeled with the *retarget* class should be resized in a content-aware fashion by the servlet and reloaded onto the webpage. This behavior can be obtained using a set of javascript functions which, registering themselves (using closures) to the stack of *onresize* and *onload* callbacks, when executed will:
- parse the HTML page DOM looking for images labaled with the *retarget* class;
- edit their *src* attribute with the correct servlet request;
- display the new image into the page.
All of these functions are contained in a single javascript file, named *retarget.js* which should be included in the *<HEAD>* section of the HTML document. In this way a webpage using content-aware resized liquid images will look like the following:

```
<html>
  <head>
    ...
    <script language="javascript1.5" src="retarget.js">
    </script>
    ...
  </head>
  <body>
  ...
    <img class="retarget" src="path_to_img.jpg" width="20%" />
  ...
  </body>
</html>
```

In this way, almost no effort at all should be done by the user, which just needs to include a javascript file and add a class label to the images he or she wants to resize in a content-aware fashion.

The described system was tested with several display devices and different browsers in order to assure complete cross-platform and cross-device compliance. In particular the tests were done on the following browsers:

Figure 6.6: System at work: in (a) desktop computer browser, canvas width = 1270px, top right original image, top left homogeneous resizing, bottom left content-aware resizing. In (b) and (c), mobile phone browser, canvas width = 320px, top homogeneous resizing, bottom content-aware resizing.

- Google Chrome
- Mozilla Firefox
- Microsoft Internet Explorer
- Apple Safari
- Nokia S60 OSS Mobile Browser
- Opera Mobile

The resizing is operated correctly notwithstanding the resolution or the display size used. In Figure 6.6 several example of the systems at work are shown. As can be seen in Figure 6.6a, the original image (top right) has been resized to 50% of the window size in a desktop PC web browser on a high resolution screen. On top left is shown the result of homogeneous resizing, where is evident the shrinking of the main object. On bottom left is depicted the result of content-aware resizing, the size of the main object is unchanged and the deformation are diffused into the background. In Figure 6.6b-c, the same image is resized onto the browser of a low resolution mobile device. The homogeneous resize shrinks drastically the main object, while it is kept almost unchanged in the content-aware resized version, where just the background content is lost.

Although its purpose is different from the described application, an online content-aware image resize service called *rsizr*[1] implementing seam-carving algorithm, can be used for comparing and evaluating performances. The main differences from the two approach is that seam-carving is iterative, thus its timing depends on the number of seams to remove/add. The proposed system is one-shot, it just needs one step to resize an image to an arbitrary image size. These differences emerge from the plot of timing performance depicted in Figure 6.7. The values report the processing time for resizing an image of resolution 1024x768 in terms of seconds versus the target image size. For seam-carving, the smaller the target size, the higher is the processing time, since a larger number of seams have to be removed. For the described system instead, the elapsed time keeps quite constant, getting a bit shorter as the target image size decrease. This is due to the smaller amount of data needed to be transmitted to the browser once the resizing task is complete.

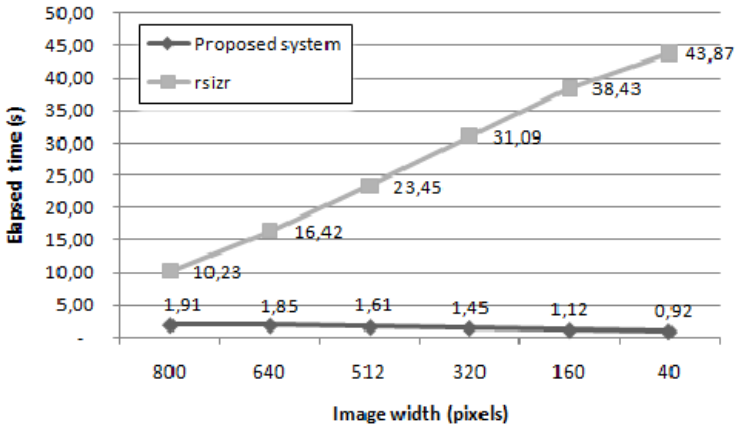


Figure 6.7: Timing performance comparisons between the proposed system and *rsizr* service based on seam carving.

[1]http://rsizr.com/

Chapter 7

Conclusions and future work

In this dissertation a theoretic framework for non-rigid image registration along with its application was presented. It relies on kernel regression and fuzzy c-means. In particular, fuzzy membership maps obtained as the result of fuzzy clustering are used as equivalent kernels for regression, allowing to reconstruct full global deformations needed to align input and target images on the basis of some prior local information. After the framework explanation, three different applications are introduced and described. Two of them (SLB and ILB) are landmark-based, while the third one (AAB) is area-based. SLB and ILB rely on manual selection or automatic extraction of some landmark points to recover the transformation in a one-shot fashion, while AAB, assuming as input just the two images, recovers the needed correspondences (and the consequent local deformations) by maximizing the normalized mutual information of the sub-regions of the images. Local transformations are subsequently composed using fuzzy kernel regression to obtain a unique global registration function. The method works both for 2D slices and 3D volumes.

Several optimizations, such as efficient operators implementation and GPGPUs parallel procedure development for Mutual Information estimation and fuzzy c-means clustering allow a remarkable speed up of the computation, essential for the huge size of medical image datasets.

After a theoric framework evaluation, its applications were tested and performance measured with several experiments. The experimentation was done taking both a quantitative and a qualitative assessment. Tests were operated on synthetic and real datasets, both mono- and multi-modal. Mono-modal tests were

conducted for intra-patient evaluation, with scans taken at different time and presenting anatomical differences. For multi-modal images different patients scans were used to align completely different anatomical structures. Finally, a team of expert radiologists performed evaluation of real mono- and multi-modal pairs to assess actual application of the framework in a true diagnostic environment.
The fuzzy kernel regression framework demonstrated to be usable for other spatial image transformation purposes: the case of retargeting was presented, the proposed method achieves promising results comparable to existing literature approaches. Future work consists in the extension and improvement of the current methods and in its application to additional image processing scenarios, such as mosaicing and virtual garment.

Appendix A

GPUs and Nvidia CUDA

The following appendix contains an introduction to GPUs and CUDA devices, and is mainly an excerpt from Nvidia CUDA Programming Guide, NVIDIA (2008).

A.1 Graphics Processing Units

A graphics processing unit or GPU is a specialized microprocessor that offloads and accelerates graphics rendering from the central (micro-)processor. Currently, it is often used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics, and their highly parallel structure makes them more effective than general-purpose CPUs for a range of data-parallel algorithms. In a personal computer, a GPU can be present on a video card, or it can be on the motherboard, or as in certain Core Intel CPUs, on a CPU die. More than 90% of new desktop and notebook computers have integrated GPUs, which are usually far less powerful than those on a dedicated video card.
The term was defined and popularized by Nvidia in 1999, who marketed the GeForce 256 as "the world's first 'GPU', or Graphics Processing Unit, a single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that is capable of processing a minimum of 10 million polygons

per second." Rival ATI Technologies coined the term visual processing unit or VPU with the release of the Radeon 9700 in 2002.
Early GPUs main functions were strictly related to computer graphics operations, such as polygons rendering, texture mapping, vertices manipulation, oversampling and interpolation. However since, such functions act mainly on matrices and vectors, the interest for GPUs has increased with studies involving general computing. This road led to the introduction of the concept of GPGPUs (general purpose graphics processing unit). GPGPUs devices left the scheme imposed by the graphic pipeline, providing a general-purpose computing power, as opposed to being hard wired solely to do graphical operations.
This concept was mainly followed by the two main GPUs designers, Nvidia and ATI.
Recently NVidia began releasing cards supporting an API extension to the C programming language CUDA ("Compute Unified Device Architecture"), which allows specified functions from a normal C program to run on the GPU's stream processors. This makes C programs capable of taking advantage of a GPUs ability to operate on large matrices in parallel, while still making use of the CPU when appropriate. CUDA is also the first API to allow CPU-based applications to access directly the resources of a GPU for more general purpose computing without the limitations of using a graphics API.

A.2 Nvidia CUDA

Following is an excerpt from CUDA Programming Guide NVIDIA (2008).
CUDA (an acronym for Compute Unified Device Architecture) is a parallel computing architecture developed by NVIDIA. CUDA is the computing engine in NVIDIA graphics processing units (GPUs) that is accessible to software developers through variants of industry standard programming languages. Programmers use 'C for CUDA' (C with NVIDIA extensions and certain restrictions), compiled through a PathScale Open64 C compiler, to code algorithms for execution on the GPU. CUDA architecture shares a range of computational interfaces with two competitors -the Khronos Group's Open Computing Language and Microsoft's DirectCompute. Third party wrappers are also available for Python, Perl, For-

tran, Java, Ruby, Lua, MATLAB and IDL, and native support exists in Mathematica. CUDA gives developers access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs. Using CUDA, the latest NVIDIA GPUs become accessible for computation like CPUs. Unlike CPUs however, GPUs have a parallel throughput architecture that emphasizes executing many concurrent threads slowly, rather than executing a single thread very quickly. This approach of solving general purpose problems on GPUs is known as GPGPU. In the computer game industry, in addition to graphics rendering, GPUs are used in game physics calculations (physical effects like debris, smoke, fire, fluids); examples include PhysX and Bullet. CUDA has also been used to accelerate non-graphical applications in computational biology, cryptography and other fields by an order of magnitude or more. An example of this is the BOINC distributed computing client. CUDA provides both a low level API and a higher level API. The initial CUDA SDK was made public on 15 February 2007, for Microsoft Windows and Linux. Mac OS X support was later added in version 2.0, which supersedes the beta released February 14, 2008. CUDA works with all NVIDIA GPUs from the G8X series onwards, including GeForce, Quadro and the Tesla line. NVIDIA states that programs developed for the GeForce 8 series will also work without modification on all future NVIDIA video cards, due to binary compatibility.

A.2.1 CUDA programming model

When programmed through CUDA, the GPU is viewed as a compute device capable of executing a very high number of threads in parallel. It operates as a coprocessor to the main CPU, or host: In other words, data-parallel, compute-intensive portions of applications running on the host are off-loaded onto the device.

More precisely, a portion of an application that is executed many times, but independently on different data, can be isolated into a function that is executed on the device as many different threads. To that effect, such a function is compiled to the instruction set of the device and the resulting program, called a kernel, is downloaded to the device.

Both the host and the device maintain their own DRAM, referred to as host mem-

ory and device memory, respectively. One can copy data from one DRAM to the other through optimized API calls that utilize the devices high-performance Direct Memory Access (DMA) engines.
The batch of threads that executes a kernel is organized as a grid of thread blocks as illustrated in Figure A.1. A thread block is a batch of threads that can cooperate together by efficiently sharing data through some fast shared memory and synchronizing their execution to coordinate memory accesses. More precisely, one can specify synchronization points in the kernel, where threads in a block are suspended until they all reach the synchronization point.
Each thread is identified by its thread ID, which is the thread number within the block. To help with complex addressing based on the thread ID, an application can also specify a block as a two- or three-dimensional array of arbitrary size and identify each thread using a 2- or 3-component index instead. For a two-dimensional block of size (D_x, D_y), the thread ID of a thread of index (x, y) is $(x + yD_x)$ and for a three-dimensional block of size (D_x, D_y, D_z), the thread ID of a thread of index (x, y, z) is $(x + yD_x + zD_xD_y)$.
There is a limited maximum number of threads that a block can contain. However, blocks of same dimensionality and size that execute the same kernel can be batched together into a grid of blocks, so that the total number of threads that can be launched in a single kernel invocation is much larger. This comes at the expense of reduced thread cooperation, because threads in different thread blocks from the same grid cannot communicate and synchronize with each other. This model allows kernels to efficiently run without recompilation on various devices with different parallel capabilities: A device may run all the blocks of a grid sequentially if it has very few parallel capabilities, or in parallel if it has a lot of parallel capabilities, or usually a combination of both. Each block is identified by its block ID, which is the block number within the grid. To help with complex addressing based on the block ID, an application can also specify a grid as a two-dimensional array of arbitrary size and identify each block using a 2-component index instead. For a two-dimensional block of size (D_x, D_y), the block ID of a block of index (x, y) is $(x + yD_x)$.

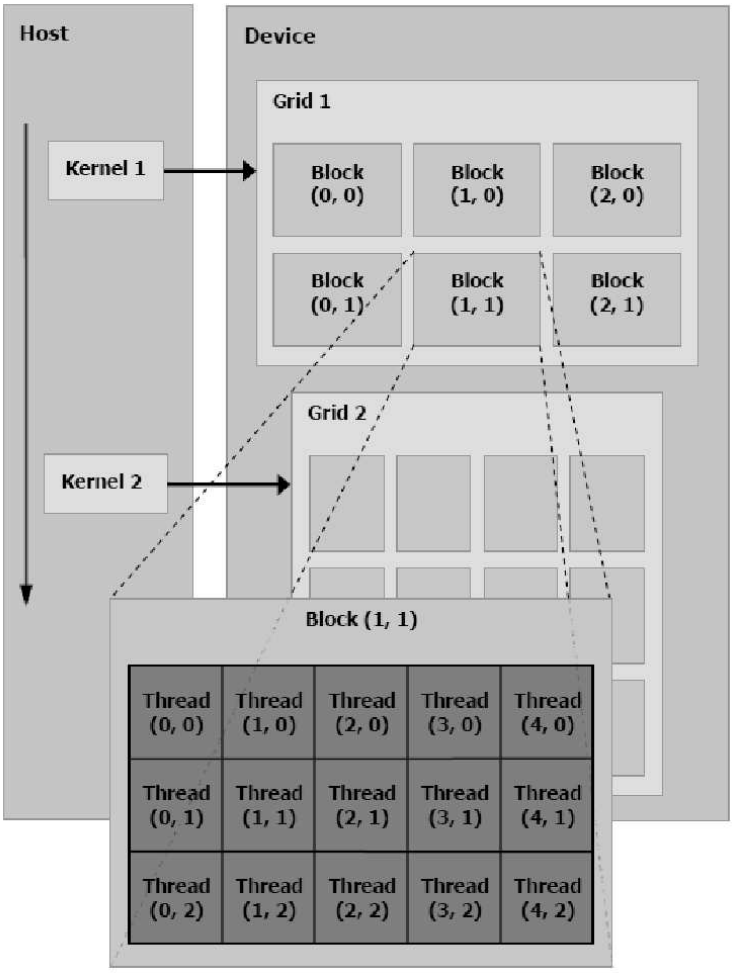


Figure A.1: The host issues a succession of kernel invocations to the device. Each kernel is executed as a batch of threads organized as a grid of thread blocks.

A.2.2 Memory model

A thread that executes on the device has only access to the devices DRAM and on-chip memory through the following memory spaces, as shown in Figure A.2:

- Read-write per-thread registers,
- Read-write per-thread local memory,

- Read-write per-block shared memory,
- Read-write per-grid global memory,
- Read-only per-grid constant memory,
- Read-only per-grid texture memory.

The global, constant, and texture memory spaces can be read from or written to by the host and are persistent across kernel launches by the same application. The global, constant, and texture memory spaces are optimized for different memory usages. Texture memory also offers different addressing modes, as well as data filtering, for some specific data formats.

A.2.3 Hardware implementation

The device is implemented as a set of multiprocessors as illustrated in Figure A.3. Each multiprocessor has a Single Instruction, Multiple Data architecture (SIMD): At any given clock cycle, each processor of the multiprocessor executes the same instruction, but operates on different data.
Each multiprocessor has on-chip memory of the four following types:

- One set of local 32-bit registers per processor,
- A parallel data cache or shared memory that is shared by all the processors and implements the shared memory space,
- A read-only constant cache that is shared by all the processors and speeds up reads from the constant memory space, which is implemented as a read-only region of device memory,
- A read-only texture cache that is shared by all the processors and speeds up reads from the texture memory space, which is implemented as a read-only region of device memory.

The local and global memory spaces are implemented as read-write regions of device memory and are not cached.

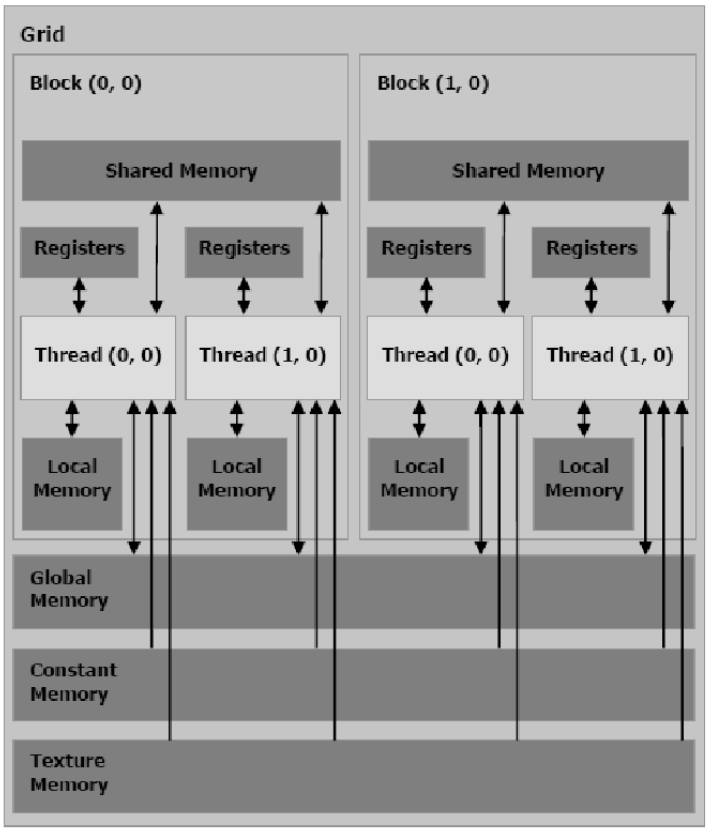


Figure A.2: A thread has access to the devices DRAM and on-chip memory through a set of memory spaces of various scopes.

Each multiprocessor accesses the texture cache via a texture unit that implements the various addressing modes and data filtering mentioned in the previous paragraph.

A.2.4 Execution model

A grid of thread blocks is executed on the device by executing one or more blocks on each multiprocessor using time slicing: Each block is split into SIMD groups of threads called warps; each of these warps contains the same number of threads, called the warp size, and is executed by the multiprocessor in a SIMD fashion;

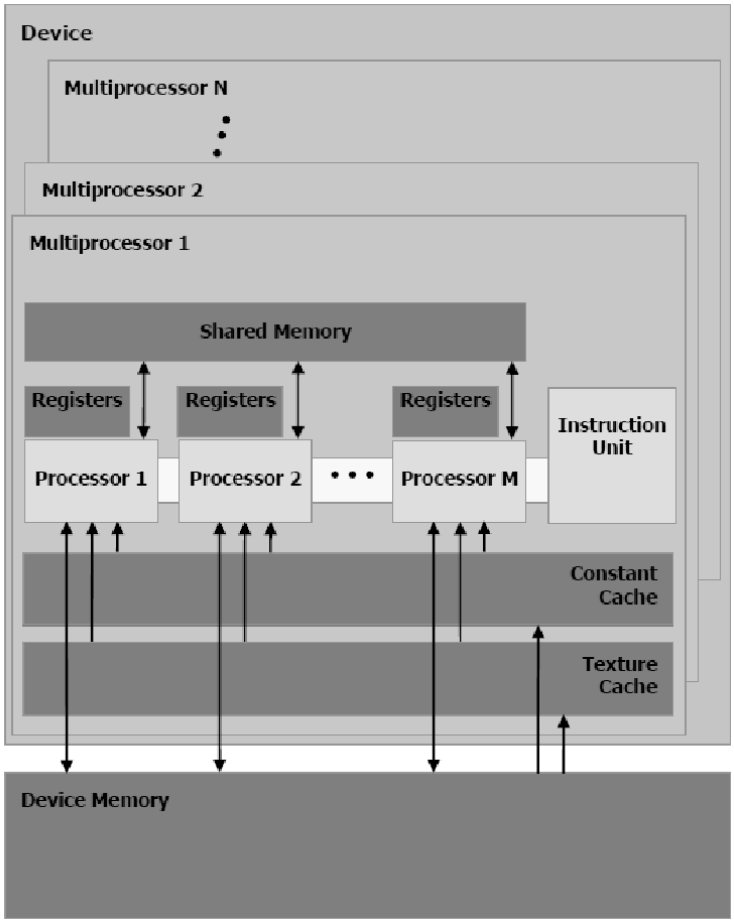


Figure A.3: A set of SIMD multiprocessors with on-chip shared memory.

a thread scheduler periodically switches from one warp to another to maximize the use of themultiprocessors computational resources. A half-warp is either the first or second half of a warp. The way a block is split into warps is always the same; each warp contains threads of consecutive, increasing thread IDs with the first warp containing thread 0.

A block is processed by only one multiprocessor, so that the shared memory space resides in the on-chip shared memory leading to very fast memory accesses. The multiprocessors registers are allocated among the threads of the block. If the number of registers used per thread multiplied by the number of threads in the

block is greater than the total number of registers per multiprocessor, the block cannot be executed and the corresponding kernel will fail to launch.
Several blocks can be processed by the same multiprocessor concurrently by allocating the multiprocessors registers and shared memory among the blocks.
The issue order of the warps within a block is undefined, but their execution can be synchronized to coordinate global or shared memory accesses.
The issue order of the blocks within a grid of thread blocks is undefined and there is no synchronization mechanism between blocks, so threads from two different blocks of the same grid cannot safely communicate with each other through global memory during the execution of the grid.
If a non-atomic instruction executed by a warp writes to the same location in global or shared memory for more than one of the threads of the warp, the number of serialized writes that occur to that location and the order in which they occur is undefined, but one of the writes is guaranteed to succeed. If an atomic instruction executed by a warp reads, modifies, and writes to the same location in global memory for more than one of the threads of the warp, each read, modify, write to that location occurs and they are all serialized, but the order in which they occur is undefined.

A.3 Application Programming Interface

The goal of the CUDA programming interface is to provide a relatively simple path for users familiar with the C programming language to easily write programs for execution by the device. It consists of:

- A minimal set of extensions to the C language, described in Section 4.2, that allow the programmer to target portions of the source code for execution on the device;
- A runtime library split into:
  - A host component, described in Section 4.5, that runs on the host and provides functions to control and access one or more compute devices from the host;

– A device component, described in Section 4.4, that runs on the device and provides device-specific functions;
– A common component, described in Section 4.3, that provides built-in vector types and a subset of the C standard library that are supported in both host and device code.

References

Anderson, D.T., III, R.H.L. & Keller, J.M. (2008). Speedup of fuzzy clustering through stream processing on graphics processing units. *IEEE T. Fuzzy Systems*, **16**, 1101–1106. 41

Arsigny, V., Pennec, X. & Ayache, N. (2003). Polyrigid and polyaffine transformations: A new class of diffeomorphisms for locally rigid or affine registration. In *MICCAI (2)*, 829–837. 11

Avidan, S. & Shamir, A. (2007). Seam carving for content-aware image resizing. *ACM Trans. Graph.*, **26**, 10. 60, 62, 63

Bajcsy, R. & Kovačič, S. (1989). Multiresolution elastic matching. *Comput. Vision Graph. Image Process.*, **46**, 1–21. 12

Bezdek, J.C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms (Advanced Applications in Pattern Recognition)*. Springer. 18

Blinn, J. (1998). *Jim Blinn's corner: dirty pixels*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 35

Bookstein, F.L. (1989). Principal warps: thin-plate splines and the decomposition of deformations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **11**, 567–585. 11

Brown, L.G. (1992). A survey of image registration techniques. *ACM Computing Surveys*, **24**, 325–376. 11

Cocosco, C.A., Kollokian, V., s. Kwan, R.K. & Evans, A.C. (1997). Brainweb: Online interface to a 3d mri simulated brain database. *NeuroImage*, **5**, 425. 48

Collins, D.L., Zijdenbos, A.P., Kollokian, V., Sled, J.G., Kabani, N.J., Holmes, C.J. & Evans, A.C. (1998). Design and construction of a realistic digital brain phantom. *IEEE Trans Med Imaging*, **17**, 463–468. 48

Cover, T.M. & Thomas, J.A. (1991). *Elements of information theory*. Wiley-Interscience, New York, NY, USA. 35

Delaunay, B.N. (1934). Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, 793–800. 24

Fornefett, M., Rohr, K., Stiehl, H.S. & Systeme, A.K. (2001). Radial basis functions with compact support for elastic registration of medical images. *IVC*, **19**, 1–2. 11

Gaens, T., Maes, F., Vandermeulen, D. & Suetens, P. (1998). Non-rigid multimodal image registration using mutual information. In *MICCAI '98: Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention*, 1099–1106, Springer-Verlag, London, UK. 12

Gal, R., Sorkine, O. & Cohen-Or, D. (2006). Feature-aware texturing. In *Proceedings of Eurographics Symposium on Rendering*, 297–303. 60, 62

Harris, C. & Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, 147–151. 61

Hawkes, D.J. (1998). Algorithms for radiological image registration and their clinical application. *Journal of Anatomy*, 347–361. 11

Itti, L., Koch, C. & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 1254–1259. 61, 63

Ji, X., Pan, H. & Liang, Z.P. (1999). A region-based mutual information method for image registration. In *The 7th Scientific Meeting Exhibition of the ISMRM*. 32

Ji, X., Pan, H. & Liang, Z.P. (2003). Further analysis of interpolation effects in mutual information-based image registration. *IEEE Trans. Med. Imaging*, **22**, 1131–1140. 12, 33, 34, 35, 52, 55

Kohlrausch, J., Rohr, K. & Stiehl, H.S. (2005). A new class of elastic body splines for nonrigid registration of medical images. *J. Math. Imaging Vis.*, **23**, 253–280. 29

Kwan, R.K.S., Evans, A.C. & Pike, G.B. (1996). An extensible mri simulator for post-processing evaluation. In *VBC '96: Proceedings of the 4th International Conference on Visualization in Biomedical Computing*, 135–140, Springer-Verlag, London, UK. 48

Kwan, R.K.S., Evans, A.C. & Pike, G.B. (1999). Mri simulation-based evaluation of image-processing and classification methods. *Medical Imaging, IEEE Transactions on*, **18**, 1085–1097. 48

Lester, H. & Arridge, S.R. (1999). A survey of hierarchical non-linear medical image registration. *Pattern Recognition*, **32**, 129–149. 11

Likar, B. & Pernus, F. (2001). A hierarchical approach to elastic registration based on mutual information. *Image Vision Comput.*, **19**, 33–44. 11

Liu, C., Yuen, J., Torralba, A., Sivic, J. & Freeman, W.T. (2008). SIFT Flow: dense correspondence across different scenes. In *ECCV*. 67

Liu, F. & Gleicher, M. (2006). Video retargeting: automating pan and scan. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, 241–250, ACM, New York, NY, USA. 61

Lomont, C. (2003). Understanding quake's fast inverse square root. 44

Maes, F., Collignon, A., Vandermeulen, D., Marchal, G. & Suetens, P. (1996). Multi-modality image registration maximization of mutual information. In *MMBIA '96: Proceedings of the 1996 Workshop on Mathematical*

Methods in Biomedical Image Analysis (MMBIA '96), 14, IEEE Computer Society, Washington, DC, USA. 11, 36

Maes, F., Collignon, A., Vandeermeulen, D., Marchal, G. & Suetens, P. (1997). Multimodality image registration by maximization of mutual information. *IEEE Transactions in Medical Imaging*, **16**, 187–198. 33, 35, 37

Maintz, J. & Viergever, M. (1998). A survey of medical image registration. *Medical Image Analysis*, **2**, 1–36. 11

Maintz, J.B.A., Meijering, E.H.W. & Viergever, M.A. (1998). General multimodal elastic registration based on mutual information. In *Image Processing*, 144–154, SPIE Press. 12

Marcus, D.S., Wang, T.H., Parker, J., Csernansky, J.G., Morris, J.C. & Buckner, R.L. (2007). Open access series of imaging studies (oasis): Cross-sectional mri data in young, middle aged, nondemented, and demented older adults. *J. Cognitive Neuroscience*, **19**, 1498–1507. 48

Mehrotra, S. (1992). On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, **2**, 575–601. 64

Meyer, C.R., Boes, J.L., Kim, B., Bland, P.H., R, K., Zasadny, Kison, P.V., Koral, K., Frey, K.A. & Wahl, R.L. (1997). Demonstration of accuracy and clinical versatility of mutual information for automatic multimodality image fusion using affine and thin-plate spline warped geometric deformations. *Medical Image Analysis*, **1**, 195–206. 11

Moddemeijer, R. (1989). On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, **16**, 233–246. 36

Nadaraya, E.A. (1964). On estimating regression. *Theory of Probability and its Applications*, **9**, 141–142. 13

Nielsen, B.M. & Gramkow, C. (1996). Fast fluid registration of medical images. vol. 1131, 267–276. 12

NVIDIA (2008). *NVIDIA CUDA Programming Guide 2.0*. 75, 76

Ourselin, S., Roche, A., Prima, S. & Ayache, N. (2000). Block matching: A general framework to improve robustness of rigid registration of medical images. In A. DiGioia & S. Delp, eds., *Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery (MICCAI 2000)*, vol. 1935 of *Lectures Notes in Computer Science*, 557–566, Springer, Pittsburgh, Penn, USA. 12

Pele, O. & Werman, M. (2009). Fast and robust earth mover's distances. In *ICCV*. 67

Pluim, J.P.W. & Fitzpatrick, J.M. (2003). Image registration. *IEEE Trans. Med. Imaging*, **22**, 1341–1343. 11

Pluim, J.P.W., Maintz, J.B.A. & Viergever, M.A. (2000). Interpolation artefacts in mutual information-based image registration. *Comput. Vis. Image Underst.*, **77**, 211–232. 12, 32, 33, 34, 36

Rubinstein, M., Shamir, A. & Avidan, S. (2008). Improved seam carving for video retargeting. *ACM Trans. Graph.*, **27**, 1–9. 60, 62, 63, 66

Rubinstein, M., Shamir, A. & Avidan, S. (2009). Multi-operator media retargeting. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2009)*, **28**, 1–11. 61, 62, 66

Rubinstein, M., Gutierrez, D., Sorkine, O. & Shamir, A. (2010). A comparative study of image retargeting. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, **29**. 66

Santella, A., Agrawala, M., Decarlo, D., Salesin, D. & Cohen, M. (2006). Gaze-based interaction for semi-automatic photo cropping. In *In CHI 2006*, 771–780. 61

Studholme, C. (1999). An overlap invariant entropy measure of 3d medical image alignment. *Pattern Recognition*, **32**, 71–86. 11

Suh, B., Ling, H., Bederson, B.B. & Jacobs, D.W. (2003). Automatic thumbnail cropping and its effectiveness. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, 95–104, ACM, New York, NY, USA. 61

Tsao, J. (1999). Efficient interpolation for clustering-based multimodality image registration. 32

Tsao, J. (2003). Interpolation artifacts in multimodality image registration based on maximization of mutual information. *IEEE Transactions on Medical Imaging*. 12, 33, 52, 53

Viola, P. & Jones, M. (2001). Robust real-time object detection. In *International Journal of Computer Vision*. 61

Viola, P.A. (1995). Alignment by maximization of mutual information. 11, 36

Watson, G.S. (1964). Smooth regression analysis. *Sankhyā Ser.*, **26**, 359–372. 13

Wells, W., Viola, P., Atsumi, H., Nakajima, S. & Kikinis, R. (1996). Multi-modal volume registration by maximization of mutual information. 36

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, **4**, 389–396. 11

Wolf, L., Guttmann, M. & Cohen-Or, D. (2007). Non-homogeneous content-driven video-retargeting. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07)*. 60, 62, 63, 66

Yu-Shuen Wang, O.S., Chiew-Lan Tai & Lee, T.Y. (2008). Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA*, **27**. 61, 62, 63, 66

Zitová, B. & Flusser, J. (2003). Image registration methods: a survey. *Image Vision Comput.*, **21**, 977–1000. 11