Quantifying Skin Lesion Change over a Short Period

Cristian C Cobzarenco

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2013

Abstract

This paper presents an approach to measuring changes in skin lesions over time using a pair of colour images taken a short of time apart, providing a first step towards incorporating information about lesion evolution in automated skin lesion classification methods. The problem is split into three different tasks: image alignment, dense deformable registration (DDR) and measuring changes in pigmentation. For the image alignment task the well known technique of SIFT matching with RANSAC was used and evaluated. For DDR, a novel approach is introduced and compared against an established DDR method and a baseline. The evaluation is performed on three synthetic test sets with increasing levels of distortion created by deforming images of real skin lesions. The results of the experiments show the new DDR method is comparable in performance to the established DDR approach in general and outperforms it on mid-level deformations specifically.

Acknowledgements

First and foremost, I need to thank my supervisor Robert Fisher for his support, academically and morally, as well as for his ability to put up with my disastrous time management skills. The following people also have my gratitude as this project owes its existence to them, at least in part: the anonymous volunteers who accepted to have their lesions photographed, the thousands of kind souls who make code freely available online through the MATLAB File Exchange, the developers of the open-source library vlFeat which has been heavily used throughout the development of this project, the StackOverflow community, M. Cobzarenco and B. Mitrović for enlightening discussions which led to many of the ideas incorporated into this work and S. R. for keeping me sane in the time I was not working.

Table of Contents

Chapter 1

Introduction

1.1 Motivation and Background

In recent years there has been an increasing interest in automated skin lesion classification in the medical image analysis community, as an aid in the diagnosis of skin cancer – one of the most common malignancies in fair-skinned populations, with a documented increase in incidence in the last 50 years [5]. What makes the idea of automated diagnosis so important is that process the process of human visual diagnosis is still as much art as it is science; objective methods such as the ABCDE[1] for melanoma appearing to be less relevant than the expert's experience [1].

The existing literature on the subject of automated skin lesion classification tends to deal with the analysis and classification of a single colour picture of the lesion [8], potentially captured through a dermatoscope – a small device which magnifies, illuminates and removes specular reflections using a polarised filter. Recently a depth channel has also been used in addition to the colour image, succesfully improving both classification performance [2] and segmentation accuracy [6].

A line of research which has not been explored to the same extent is whether multiple images taken over time could be used to improve the performance of classifiers. The changes in lesion shape and colour over time could provide useful information to be used as additional features in an existing classifier. The only precedent we could find in recent literature was [9] where deformation and pigmentation changes in melanocytic lesions were manually observed and analysed using dermatoscopy with promising results. We were not, however, able to find any work on an automated diagnosis approach which incorporates this kind of information. A potential explanation for this lack of research is that gathering such data is problematic: lesions with a high likelihood of malignancy will most often be excised as soon as possible, on the order of days at most, without providing the opportunity to take a follow-up picture of the lesion after a longer interval of time. However, in cases where lesions are not life threatening or where a diagnosis of malignancy seems very unlikely at a given point in time, having

[1] A mnemonic for features to check when diagnosing melanoma: **A**symmetry, **B**orders, **C**olor, **D**iameter and **E**volving.

access to a method to analyse a lesion's evolution since the patient's last visit could prove valuable, be it as part of a fully automated diagnosis tool or only as an aid to traditional diagnosis.

The first step in building such a tool or augmenting an existing classifier is of course having an approach for quantifying changes in shape and colour. In the following section we summarise our proposed approach to this problem, a collection of methods to deal separately with alignment, deformation and colour.

1.2 Contributions

The proposed approach analyses the differences between two colour pictures of a lesion captured during two different sessions separated by a short period of time (on the order of a month). We assume that the conditions in which the photographs are take are reasonably controlled, particularly that there are no major changes in lighting. We also assume that the lesion has been previously isolated within the pictures by some other means – a process called **segmentation** – giving the proposed algorithms access to a binary mask which labels each pixel as either 'healthy' or 'lesion'. Figure 1.1 shows an example of two such pictures and their masks. The **data acquisition** process is described in more detail in §3.1.1.

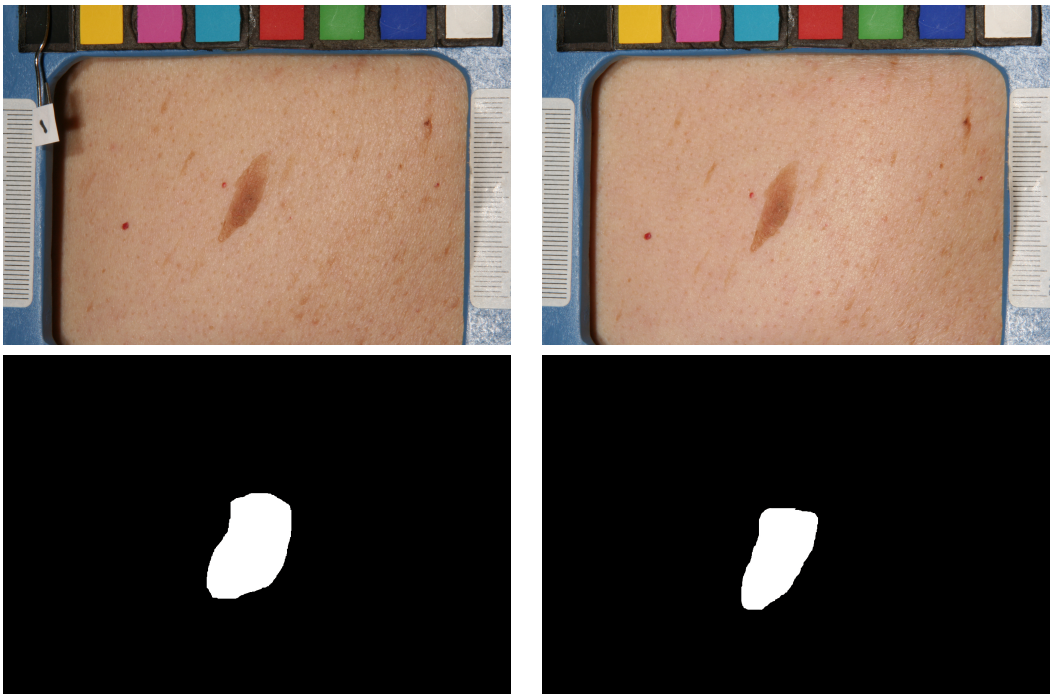


Figure 1.1: An example of two pictures of the same skin lesion taken 38 days apart. The second row shows the binary masks which highlight the lesion. In general we refer to the first image chronologically as the 'source' image, and to the second one as the 'target'. The blue frame is part of the rig used to capture the photographs (§3.1.1).

1.2.1 Image Alignment and Colour Normalisation

The first step in measuring the changes in the lesion's shape and colour is to remove any differences between the two images which are due to different view angles (pose) or lighting, to avoid reporting spurious changes. The images are first **spatially aligned** using a well-known technique which relies on estimating and matching 'points of interest' between the two images. By analysing the difference in screen position between matching points of interest, we can estimate a transformation which aligns the two images approximately in the same space. This process uses the SIFT algorithm [7] to generate points of interest and a RANSAC method to match them. A detailed description is given in §2.2.

To correct for changes in **lighting**, a simpler method is employed: the colours are corrected using a healthy skin patch around the lesion, ensuring the patch has the same colour mean and standard deviation in both. This process is described in §2.1.

1.2.2 Measuring Changes

The most important measure we consider is the way in which persistent morphological features[2] of the lesion move relative to each other. This is quantified as a 'displacement field', a mapping between pixels of the target image to vectors encoding the amount and direction of movement w.r.t. the source image. This problem is generally called **'dense deformable registration'** (DDR), a well-known and well-researched problem in the field of medical image analysis [3]. We introduce a novel DDR method, called 'probabilistic multiscale registration' (PMR) and compare it against an established method as well as two other, less sophisticated methods.

The simplest DDR method we introduce, called **'enhanced least squares matching'** (or ELSM), relies on a search for each pixel from the target image inside the source image by comparing pixel neighbourhoods. This method suffers from a series of issues described in §2.3.2 which are individually addressed by implementing a series of small improvements, thus 'enhancing' the basic method. Because it estimates each vector independently, the resulting approach still suffers from two major issues: estimated fields may contain self-intersections (the field may fold onto itself) and large scale deformation cannot be reliably registered.

A more significant change is required to address these problems: the **Multiscale ELSM** algorithm (MS-ELSM, described in §2.3.3) performs the ELSM registration progressively on different scales and combines the resulting estimates into a final displacement field estimate. Since at each scale the norm of the vectors can be limited to at most half the distance between them, the resulting field is guaranteed not to contain any self-intersections. The MS-ELSM method suffers from a more subtle issue, however, in areas of low contrast such as uniform patches of skin matching cannot be performed reliably (imagine a flat colour image – displacement vectors can be interchanged freely

[2] 'persistent' refers to features which exist in both the source and target image.

with no increase in error). In such regions MS-ELSM assumes no movement has occured, which may not necessarily be the case. Ideally we would like to model the fact that the matchings cannot be estimated correctly in an area and try to infer the displacement field by interpolating between adjacent areas in which matchings are more certain. After all, if the borders of a flat colour region all move in same direction, it seems reasonable to assume the the pixels inside the region are also moving in that direction.

To address this issue, a redesign of the core ELSM algorithm is proposed for the **'probabilistic multiscale registration'** (PMR) algorithm, described in §2.3.4. When estimating displacement, rather than selecting the best match between source and target pixels, we estimate a 'matching distribution' over which source pixels are likely to match a given target pixel, taking the so-called Bayesian approach. Furthermore, we introduce a smoothing constraint as a distribution over a displacement vector given its neighbours. To maximise the *a posteriori* probability a very simple method is used. Intuitively: at each iteration, each displacement vector is 'pulled' in alignment to its neighbours, but vectors in areas of low estimated variance (high matching confidence) move more slowly than the ones in areas of high variance (low matching confidence). This heuristic deals well with areas of low contrast, since the the matching distributions of the displacement vectors in such regions will have very high variance, moving very eagerly to align with their neighbours, allowing confident matches (with low variance) around the edges to propagate inside. This method is described in §2.3.4.2.

Once persistent features are matched between the source and target images, by distorting the source image to match the target, we can compare pixel for pixel how the lesion **pigmentation** changed between the two images – note that small changes in lighting and skin colour have already been accounted for in the colour normalisation step.

1.2.3 Evaluation

Evaluation is performed separately for image alignment and deformable registration through a series of **experiments** on three different synthetic test sets with increasing levels of deformation. The process for generating this dataset is described in §3.2.1.

The **deformable registration** algorithms are trained once on a smaller training set containing images of all three levels of deformation. Then displacement fields are estimated for the test set by each of the three methods introduced, as well as by a fourth established DDR method called **MRF-FastPD** [4] a short introduction of which is given in §2.3.5. The estimates are then compared to the ground truth using several metrics and the results are reported in §3.2.3. The results suggest that the PMR algorithm performs comparably or better in cases of high deformation, while MRF-FastPD outperformed the PMR algorithm on the dataset with the least amount of deformation.

The **image alignment method** is evaluated on the same dataset, by applying a random affine transformation to each target image. Performance is measured by comparing the estimated affine transformation to ground-truth transformation. Note that since we apply the transformation to the *target* image, and then we attempt alignment with the

untransformed *source* image – the fact that the two images being aligned can be very dissimilar will cause issues with any alignment method.

There is no meaningful way of evaluating the **colour normalisation** or **pigmentation change estimation** methods on a synthetic dataset.

Chapter 2

Methods

2.1 Colour Normalisation

To correct for changes in lighting and natural differences in skin colour over time, such as those due to tanning, we employ a colour normalisation step before performing alignment and deformable registration. The method used is very simple. By morphologically dilating the lesion masks of the two images and subtracting the original masks, we can mask out a patch of healthy skin around the lesion in each of the images. Since this patch will be relative to the lesion's position, it should correspond roughly to the same region of skin in both images. See Figure 2.1 for an example.

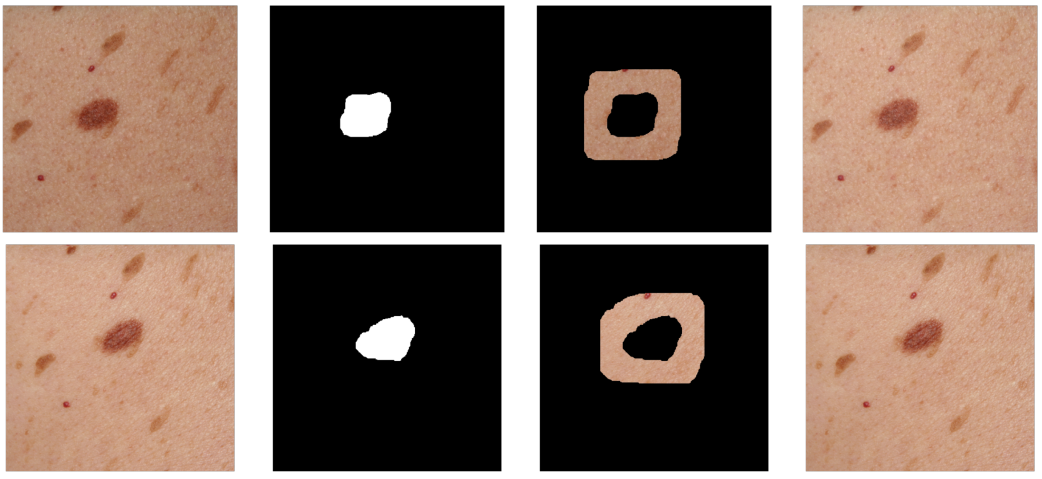


Figure 2.1: The colour normalisation process.

First the RGB means (μ_S, μ_T for the source and target images), respectively and standard deviations (σ_S, σ_T) of the skin patch in each of the images are computed. To correct an image A such that that the skin patch will have a given mean and standard deviation, we use this simple function (per colour component of each pixel in the entire

image):

$$correct(x, \mu_{\text{old}}, \sigma_{\text{old}}, \mu_{\text{new}}, \sigma_{\text{old}}) = \frac{\sigma_{\text{new}}}{\sigma_{\text{old}}} (x - \mu_{\text{old}}) + \mu_{\text{new}} \tag{2.1}$$

We apply this correction to the image with the lower standard deviation (lower contrast), because a correction from higher to lower contrast may lose information. See the last column of Figure 2.1 for the result of this process.

2.2 Image Alignment

The source and target images used to perform the registration are manually taken photographs of a moving, living, breathing patient, so it would be unreasonable to expect perfect control of the position and angle of the camera in sessions months apart. In the case of the data we use for our experiments, the camera was attached to a rigid frame (see Figure 1.1 for images of the frame; in general the frame is cropped out of the images we will be looking at) pressed against the skin of the patient to control both distance and angle, but, even so, slight differences in perspective can be noticed even with the naked eye (consider the examples from Figure 2.1).

It is important to avoid registering the difference in pose as a change in lesion shape and, as such, a preprocessing step which aligns the two images needs to be introduced. More formally, this procedure needs to estimate a homography[1] between the two images, ensuring that, after a transformation applied to either one of the images, the lesions have approximately the same orientation, position and scale.

2.2.1 Point of Interest Generation using SIFT

A straightforward path towards estimating this homography is to attempt to match points of 'interest' between the two images, such as corners, edge fragments or, the method we opted for, SIFT (Scale Invariant Feature Transform) points [7]. This approach is convenient since given a set of pairs of matching points, estimating the homography is reduced to a simple least squares problem.

First introduced by David Lowe in 1998, the Scale Invariant Feature Transform (or SIFT) is a commonly used technique for generating points of interest that are invariant to scale and stable under noise, differences in lighting and small amounts of rotation [7]. The method produces a set of points for a given image which, in addition to their two spatial coordinates, are also described by a 128-dimensional vector (a *descriptor*) which, in an intuitive sense, describes the visual characteristics of the area to which the points correspond – crucially, two visually similar areas will have similar values in their descriptors.

[1] A *homography* is an invertible linear transformation which captures, in essence, the relation between points on the same plane viewed from different angles. Assuming the height of a lesion is nearly zero, the photographed skin patch is such a plane of which the source and target images are two different views – thus linked by a homography.

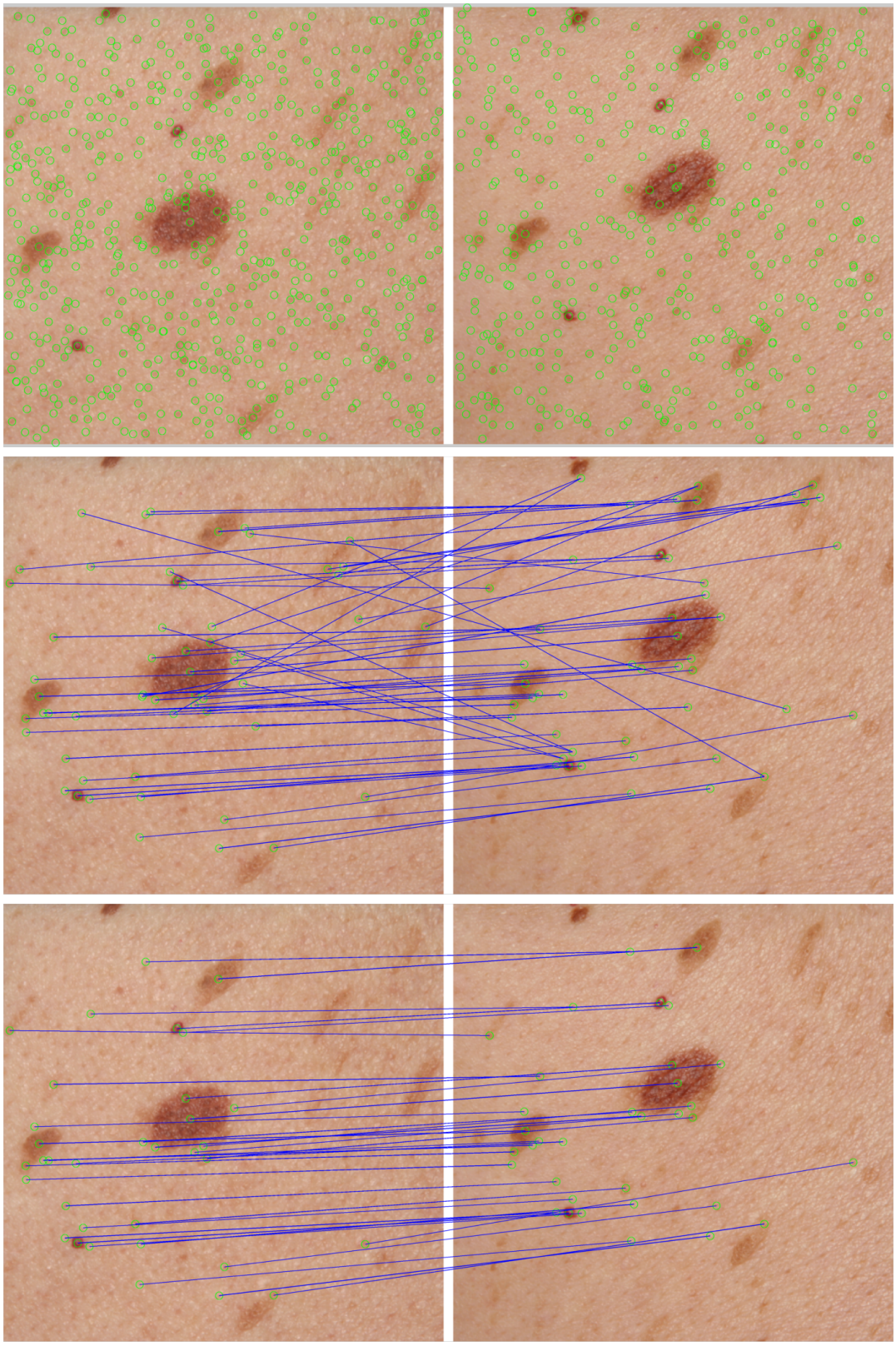


Figure 2.2: Top: all of the SIFT features generated for the source and target images (left and right respectively). Middle: the set of candidate matches C; it can be seen that while some of them are accurate, many of them are not. Bottom: the 'good set' of matches, $\mathcal{G}(H)$, obtained after initial homography estimation.

2.2.2 Candidate Matching

Before describing the matching process, we first need to introduce some notation: let the positions of the SIFT points in the source and target images be $\mathbf{s}^{(p)}_{i=1...N_s}$ and $\mathbf{t}^{(p)}_{i=1...N_t}$ respectively and, similarly, their respective 128-dimensional descriptors $\mathbf{s}^{(d)}_{i=1...N_s}$ and $\mathbf{t}^{(d)}_{i=1...N_t}$.

After the SIFT features are generated, a candidate set of pairs of matched points is created by pairing each descriptor with its nearest-neighbour – in the 128-dimensional descriptor space – from the other image. This simple, commonly used procedure is described by the following outline:

1. Initialise two sets of indices, one corresponding to the SIFT features in the source image ($I_s = \{1,2,...N_s\}$) and one to those in the target image ($I_t = \{1,2,...N_t\}$). Initialise the candidate set to the empty set ($C = \varnothing$).

2. Find the pair of source-target features $(i_0, j_0) \in I_s \times I_t$ that have the minimal Euclidean distance in descriptor space:

$$(i_0, j_0) = \underset{(i,j)\in I_s \times I_t}{\arg\min} \left\| \mathbf{s}^{(d)}_i - \mathbf{t}^{(d)}_i \right\| \tag{2.2}$$

3. Add the pair of indices to the candidate set and remove it from the two indices from the index sets:

$$C \leftarrow C \cup \{(i_0, j_0)\}, \tag{2.3}$$
$$I_t \leftarrow I_t \setminus \{i_0\} \tag{2.4}$$
$$I_s \leftarrow I_s \setminus \{j_0\} \tag{2.5}$$

4. If $I_t = \varnothing$ or $I_s = \varnothing$ then stop. Otherwise repeat from step 2.

2.2.3 Initial Homography Estimation using RANSAC

Many of the matchings generated by the algorithm described in the previous subsection will be incorrect since points inside distinct yet visually similar areas can be matched together (see the middle row in Figure 2.2). One approach to the problem of filtering out these mismatched points is a RANSAC (RANdom SAmpling Consensus) method. The main idea behind this approach is to estimate many homographies rather than a single one, by randomly selecting subsets of four candidate pairings and solving the system of equations in closed form for each subset[2]. Then, out of all these estimates,

[2] A homography is defined by a 3×3 matrix H which, for four pairings $(i_k, j_k) \in C : k = 1...4$, needs to satisfy $\exists \lambda \in \mathbb{R}^*, \forall k \in 1...4$. $H \begin{pmatrix} \mathbf{s}^{(p)}_{i_k} \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{t}^{(p)}_{j_k} \\ 1 \end{pmatrix}$. Since the equation uses homogeneous coordinates, we can set $H_{33} = \lambda$ and obtain a system of eight equations (four pairs, two dimensions) and eight unknowns (the remaining eight elements of H).

the method selects the best estimate according to a scoring metric which describes how consistent the estimated homography is with the rest of the matchings.

To understand how such a metric could be defined, first consider an ideal estimate given by the matrix H^* which is precisely correct. If there were no spurious matches in C then the result of applying the estimated transform to any source SIFT point would be equal to its matching target SIFT point. In other words, $\forall (i,j) \in C,\ \exists \lambda \in \mathbb{R}^*$ such that:

$$H^* \begin{pmatrix} \mathbf{s}_i^{(p)} \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{t}_j^{(p)} \\ 1 \end{pmatrix} \tag{2.6}$$

This property suggests a method for constructing the scoring metric: count the number of pairs $(i,j) \in C$ for which the relation in Equation (2.6) holds approximately – the distance between the transformed source point and its corresponding target point is lower than a small tolerance, ε. More formally, the score of a homography H is given by the function $S(H)$, equal to the size of the set of correctly mapped pairs $\mathcal{G}(H)$:

$$S(H) \doteq |\mathcal{G}(H)| \tag{2.7}$$

$$\mathcal{G}(H) \doteq \left\{ (i,j) \,\middle|\, (i,j) \in C, \exists \lambda \in \mathbb{R}^*. \quad \left\| H \begin{pmatrix} \mathbf{s}_i^{(p)} \\ 1 \end{pmatrix} - \lambda \begin{pmatrix} \mathbf{t}_j^{(p)} \\ 1 \end{pmatrix} \right\| \leq \varepsilon \right\} \tag{2.8}$$

Using this measure, we can select the best homography estimate $H_0 \doteq \arg\max_H S(H)$, out of all the ones estimated through the RANSAC method.

2.2.4 Homography Refinement and Preprocessing

The RANSAC method described above uses only four pairs of matching points for each estimation and as a result, due to both sensor error and limited resolution, the estimated homography will be coarse at best. However, the algorithm does provide a filtered set $\mathcal{G}(H_0)$ of matching source-target pairs[3], reducing the problem to a simple least-squares optimisation and making an unsophisticated off-the-shelf optimisation method feasible. We used Newton's method to optimise the following eight-dimensional[4] error function J[5] to obtain the optimal homography $\hat{H}_0$:

$$J(\mathbf{h}) \doteq \sum_{(i,j) \in \mathcal{G}(H_0)} \left\| \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{s}_i^{(p)} \\ 1 \end{pmatrix} - \begin{pmatrix} \mathbf{t}_j^{(p)} \\ 1 \end{pmatrix} \right\|^2 \tag{2.9}$$

$$\hat{\mathbf{h}} \doteq \arg\max_{\mathbf{h} \in \mathbb{R}^8} J(\mathbf{h}) \tag{2.10}$$

$$\hat{H}_0 \doteq \begin{pmatrix} \hat{h}_1 & \hat{h}_2 & \hat{h}_3 \\ \hat{h}_4 & \hat{h}_5 & \hat{h}_6 \\ \hat{h}_7 & \hat{h}_8 & 1 \end{pmatrix} \tag{2.11}$$

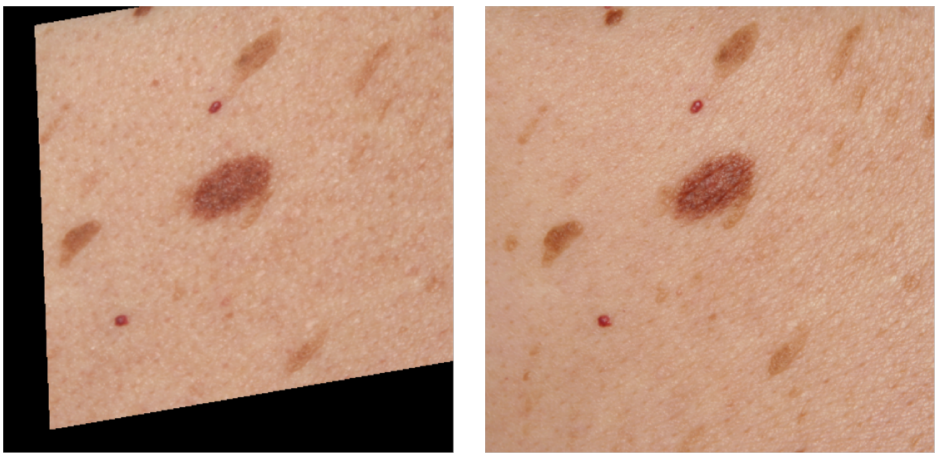


Figure 2.3: The final result of the image alignment algorithm; the lesion is in the same pose in the both images.

The transformation expressed by $\hat{H}_0$ is used to transform the source image to the space of the target image in a preprocessing step, before attempting to register changes in colour or shape. The images are also padded the same size, large enough to fit the transformed source image (see Figure 2.3). Any remaining differences between the two images are assumed to be caused by deformation in the shape of the lesion.

2.3 Measuring Changes in Shape

After the difference in pose between the two images is estimated and corrected using the method described in §2.2, deformation can be measured by analysing the remaining differences in the two images. More precisely, we are interested in modelling deformation by finding a correspondence between the pixels of the two images where there exists one, detect new growth (target pixels with no corresponding source pixels) and, potentially, remission (source pixels with no corresponding target pixels).

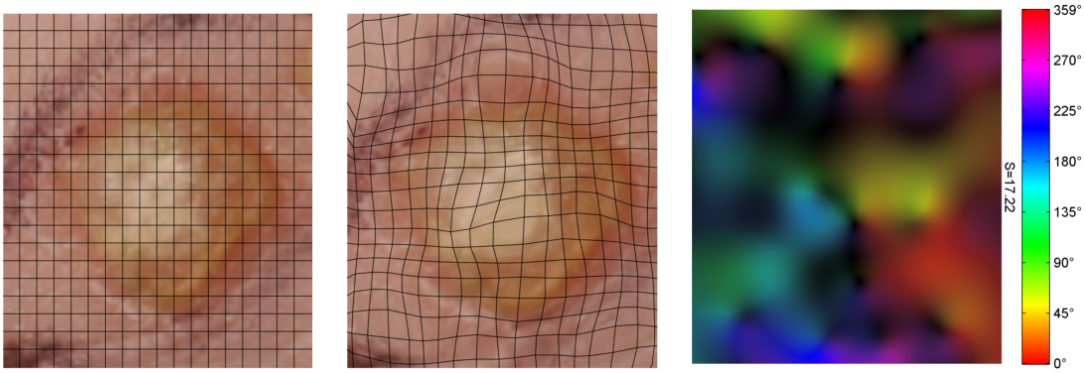
The first of these problems – finding a pixel-to-pixel correspondence between the two images to model deformation – is well-known in the medical imaging community as *dense deformable registration* (DDR) where it is commonly used for combining data from multiple sensors, or multiple captures from the same sensor during a single session to correct for the patient's breathing or any other movement.

If a DDR method also reports a probability or matching error for each pixel, this can be used for the other two problems, of detecting growth and remission: areas of growth will have high-error/low-confidence when matching from target to source, while areas of remission will have high-error/low-confidence when performing the registration in the other direction, from source to target.

[3]The 'good' set, defined in Eq 2.8, with spurious pairings removed.

[4]Since the relations use homogeneous coordinates, H_{33} can be set to 1.

[5]Sometimes called a residual in the case of a least-squares problem.



(a) (b) (c)

Figure 2.4: An example of lesion deformation, in this case generated artificially: (a) the original/source lesion image (the black grid is superimposed to help visualisation); (b) the deformed/target lesion image and (c) a visualisation of the displacement field used to produce the deformation – the hue shows the orientation of the displacement vector at a particular position, while the intensity shows its norm from zero (shown as black) to $S = 17.22$ (represented with maximal saturation and value).

In the following subsections we will first express the DDR problem more formally, introducing necessary notation, present a novel approach to deformable registration (Probabilistic Multiscale Matching, §2.3.4), as well as briefly introducing an established DDR method (MRF-FastPD in §2.3.5) and a simple baseline (Enhanced Least Square Matching, §2.3.2) against which we shall be comparing the other DDR methods.

2.3.1 Concrete Problem Setting

2.3.1.1 Images

The aligned source image S and target image T of size $w \times h$ are defined as functions from real coordinate pairs to colour red-green-blue triplets:

$$S, T \colon \mathbb{R}^2 \to [0,1]^3 \tag{2.12}$$

Any coordinate pair outside the set $[1,w] \times [1,h]$ maps to the median colour of the image by convention (the image is infinitely padded with the median) and any non-integer coordinates are resolved by through bilinear interpolation.

2.3.1.2 Displacement Fields

The task of a DDR method is to find a displacement field $D_{S \to T}$ which maps coordinate pairs in the target image to two-dimensional displacement vectors which encode

the direction and amount of movement of the pixel at the given coordinates.[6] A displacement field $D_{A\to B}$ from image A to image B, both of size $w \times h$, is formally defined as:

$$D_{A\to B}\colon [1,w]\times[1,h]\to\mathbb{R}^2 \tag{2.13}$$

Such that:

$$B(\mathbf{x}) \approx A(\mathbf{x}-D_{A\to B}(\mathbf{x})) \doteq \tilde{B}_{D_{A\to B}}(\mathbf{x}) \quad \forall \mathbf{x}\in\{1,2\ldots w\}\times\{1,2\ldots h\} \tag{2.14}$$

Where $\tilde{B}_{D_{A\to B}}\colon \mathbb{R}^2 \to [0,1]^3$ is an image of size $w \times h$, called the application of the displacement field $D_{A\to B}$ to image A, an approximation of B constructed by deforming A using the field. Non-integer coordinates are mapped using bilinear interpolation, the same as for images.

2.3.1.3 Inverse Displacement Fields

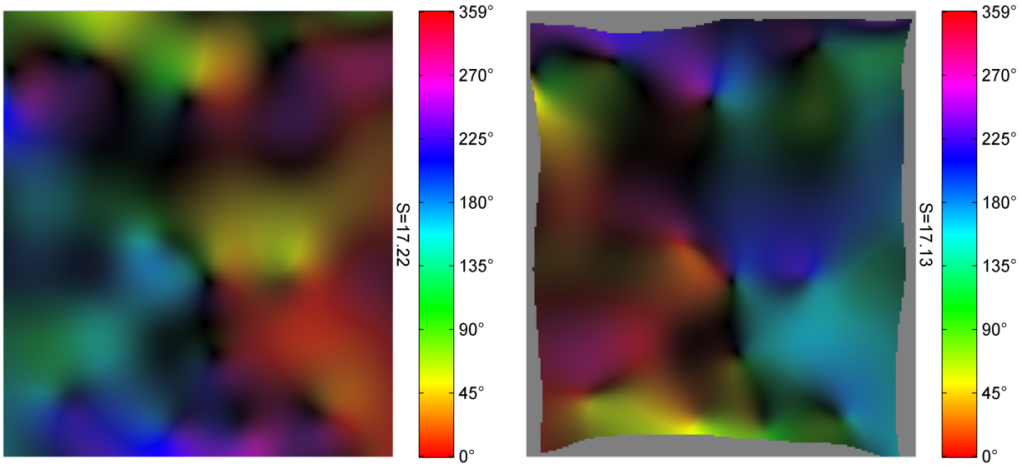


Figure 2.5: Left: An example of an artificially generated displacement field. Right: The inverse displacement field computed using Equation (2.15). The undefined region is shown in gray; it can be seen that they correspond to the areas where the original field pushed the pixels 'outside' the image.

It is natural to define an inverse displacement field $D_{B\to A}$ for a given $D_{B\to A}$ which models the deformation required to 'undistort' image B to match image A. It can be easily shown from (2.14) that a displacement field is linked to its inverse by the following relation:

$$D_{B\to A}(\mathbf{x}) = -D_{A\to B}(\mathbf{x}-D_{A\to B}(\mathbf{x})) \tag{2.15}$$

Note that the right hand side of this equation may not be defined for all $\mathbf{x}$, meaning that an inverse displacement field is potentially defined only for a region of image B. This

[6] A displacement vector $(\Delta_x,\Delta_y)^T$ at coordinates $(x,y)^T$ can be intuitively interpreted as 'the target pixel at $(x,y)^T$ has moved by $(\Delta_x,\Delta_y)^T$ to reach its current position' or, in other words, 'the target pixel at coordinates $(x,y)^T$ corresponds to the source pixel at coordinates $(x-\Delta_x,y-\Delta_y)^T$.

should be intuitive: consider, for instance, a displacement field which translates image A by 10 pixels to the right:

$$Right_{A\to B}(\mathbf{x}) = \begin{pmatrix} 10 \\ 0 \end{pmatrix} \quad \forall \mathbf{x} \in [1,w] \times [1,h] \tag{2.16}$$

Since B is of the same size as A, the 10 rightmost columns of pixels in A will be pushed 'outside' of the image and will be missing from B. Thus, the 10 rightmost columns of the inverse displacement field would have to point to the outside of the image to retrieve the missing pixels which makes them impossible to define. Figure 2.5 shows this problem when inverting a more complex field, the example from Figure 2.4.

2.3.2 Enhanced Least Squares Matching (ELSM)

The first DDR method we shall explore is 'enhanced least squares matching' or ELSM, which we shall use both as a baseline and as a part of more complex approaches. We first introduce a rudimentary method for estimating the displacement field called 'least squares matching' (LSM) in §2.3.2.1 and then build on it using several observations to construct the final ELSM algorithm.

2.3.2.1 Least Squares Matching (LSM)

One of the simplest approaches to dense deformable registration is 'least squares matching' (LSM). The main idea is to attempt to match each pixel from the target image to a pixel from the source image independently by comparing their pixel neighbourhoods. After the matchings between source and target pixels are estimated, the difference between the positions of the target pixel and the matching source pixel is reported as the displacement vector.

As mentioned before, rather than comparing individual pixel colours, we compare a neighbourhood of size[7] δ_p around each target pixel with neighbourhoods of the same size from the source image, using the sum of squared differences as a distance measure:

$$\mathrm{Nh}(\delta) \doteq \{-\delta, -\delta+1 \dots \delta\} \times \{-\delta, -\delta+1 \dots \delta\} \tag{2.17}$$

$$ssd_{A,B}(\mathbf{a},\mathbf{b}) \doteq \sum_{\mathbf{d}\in \mathrm{Nh}(\delta_p)} \|A(\mathbf{a}+\mathbf{d}) - B(\mathbf{b}+\mathbf{d})\|^2 \tag{2.18}$$

The $ssd_{A,B}(\mathbf{a},\mathbf{b})$ function can be interpreted as returning the error of matching the pixel in position $\mathbf{a}$ from image A to the pixel in position $\mathbf{b}$ in image B, taking into account a pixel neighbouhood of size δ_p. To find the best match for a pixel, the LSM method simply selects the match with the lowest error within a search window of size δ_s in the

[7] It is more convenitent to use 'half-sizes' for neighbourhoods in general: a half-size of 2 means the neighbourhood contains 2 pixels on either side of the central one, resulting in a 3×3 neighbourhood. In general, when we say a neighbourhood is of size δ, we mean it covers a region of size $(2\delta+1) \times (2\delta+1)$.

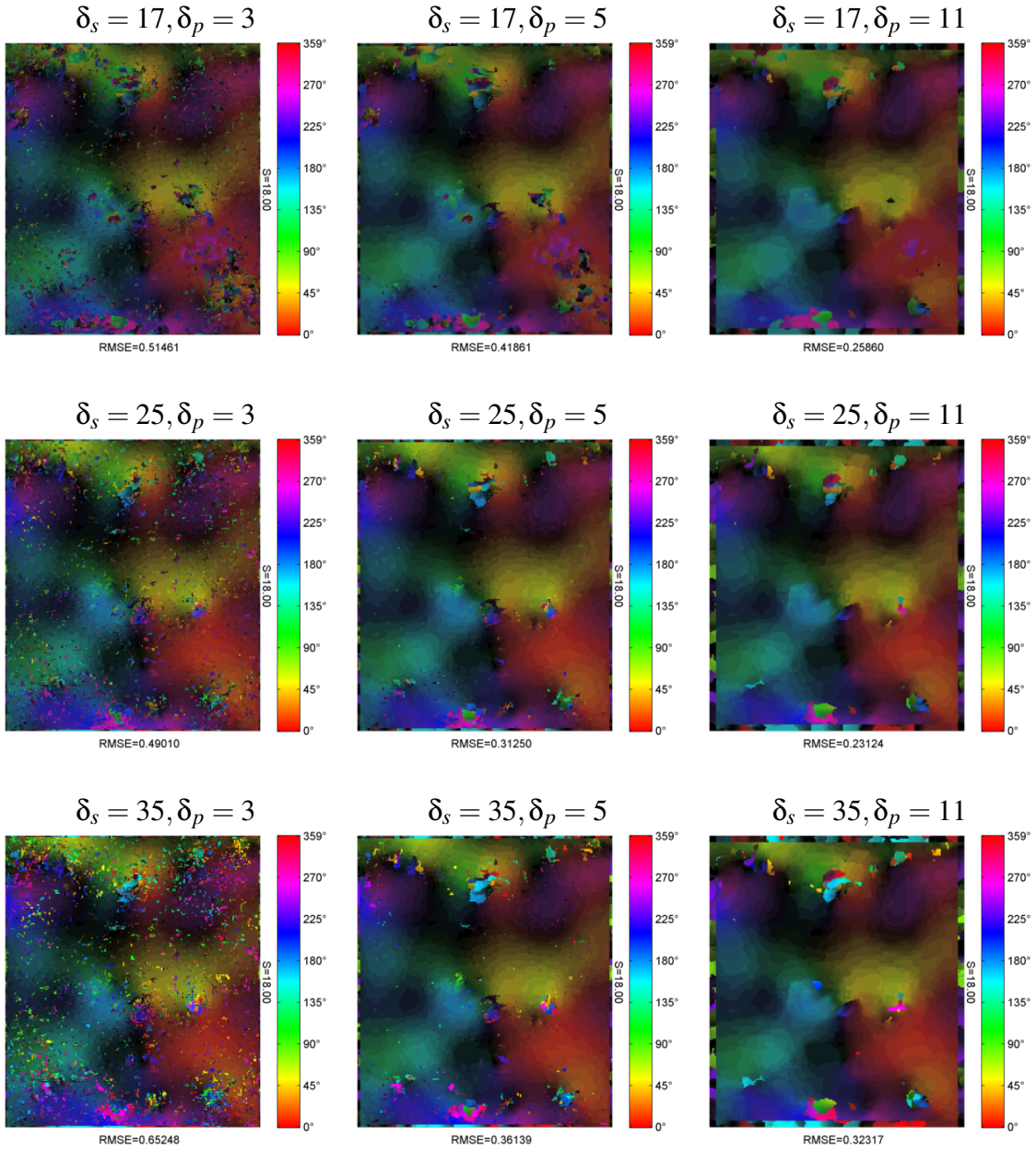


Figure 2.6: Least Squares Matching (LSM) estimates with different search window and pixel neighbourhood sizes. The root mean squared errors (RMSE) is w.r.t. the ground truth (Figure 2.4c).

source image, centered on the position of the target pixel. Formally, the displacement vector $D_{S\to T}(\mathbf{x})$ is estimated by $lsm(\mathbf{x})$, where:

$$lsm(\mathbf{x}) \doteq minssd_{S,T}(\mathbf{x}) \tag{2.19}$$

$$minssd_{A,B}(\mathbf{x}) \doteq \underset{\mathbf{d}\in \mathrm{Nh}(\delta_s)}{\arg\min}\, ssd_{A,B}(\mathbf{x}-\mathbf{d},\mathbf{x}) \tag{2.20}$$

Figure 2.6 shows a few examples of displacement estimates of the field from Figure 2.5 generated using this method for varying values of δ_p, the pixel neighbourhood size and δ_s, the search window size.

2.3.2.2 Bidirectional Search

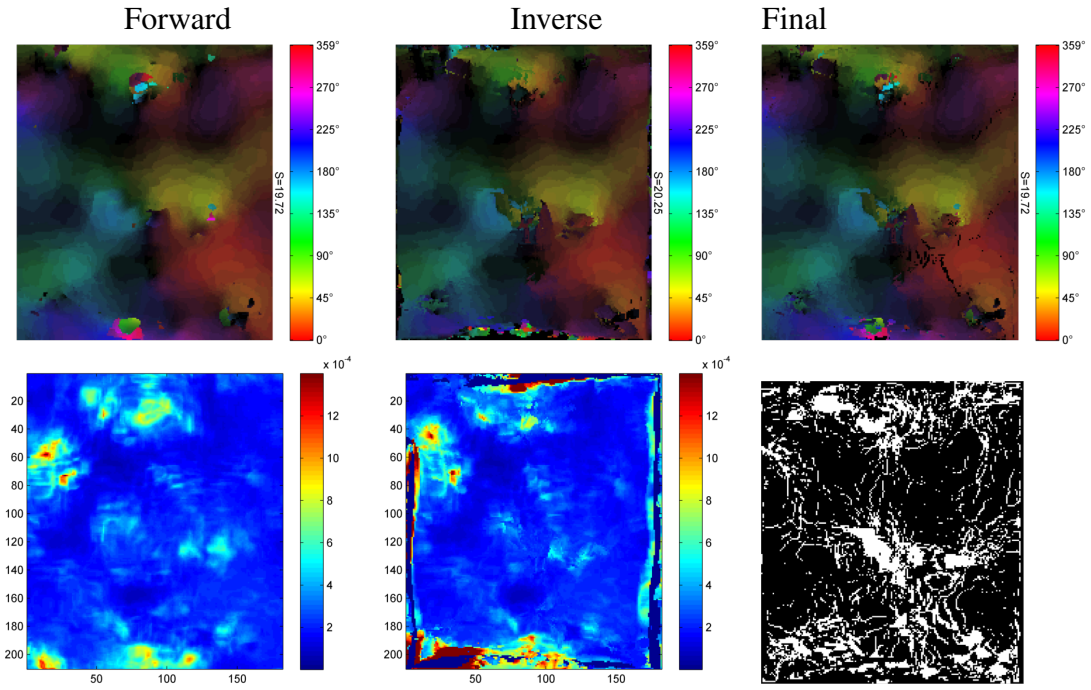


Figure 2.7: Top: the forward, inverse and displacement field estimates using bidirectional search. Bottom: the forward matching error, the inverse matching error, mask showing where inverse matching error is lower than forward matching error – this is used to create the final displacement field.

Using $minssd_{T,S}$ instead of $minssd_{S,T}$ we can also estimate the inverse displacement field which distorts the target image to match the source, rather than the other way around. Then, by applying Equation (2.15) to invert the estimate of the inverse displacement field (obtained using $minssd_{T,S}$) we obtain another estimate of the forward displacement field $lsm'(\mathbf{x})$ in addition to $lsm(\mathbf{x})$.

To obtain a better estimate of $D_{S\to T}$ we combine the two, selecting each displacement vector in the final field from the estimate with the lower *ssd* error at the given coordinates.

2.3.2.3 Distance Penalty

To model large scale deformations with the LSM method a large search window size (δ_s) is required, since the maximal norm of estimated displacement vectors is $\delta_s\sqrt{2}$. A large search window, however, also introduces a higher chance of a mismatch between distant yet visually similar areas.

To avoid this problem to a certain extent, an additional cost factor c_{dist} is introduced to penalise distant matches over closer matches with the same *ssd* matching error. ELSM uses this different error function, *essd*, which adds an additional term to the *ssd*

distance defined in Equation (2.18):

$$essd_{A,B}(\mathbf{a},\mathbf{b}) \doteq \|\mathbf{a}-\mathbf{b}\|^2 c_{\text{dist}} + \sum_{\mathbf{d}\in \text{Nh}(\delta_p)} \|A(\mathbf{a}+\mathbf{d}) - B(\mathbf{b}+\mathbf{d})\|^2 \tag{2.21}$$

The new term, $\|\mathbf{a}-\mathbf{b}\|^2 c_{\text{dist}}$, increases the *essd* cost of more distant matches, causing ELSM to prefer matches over shorter distances. The c_{dist} parameter controls the harshness of this penalty.

2.3.2.4 Contour Interpolation

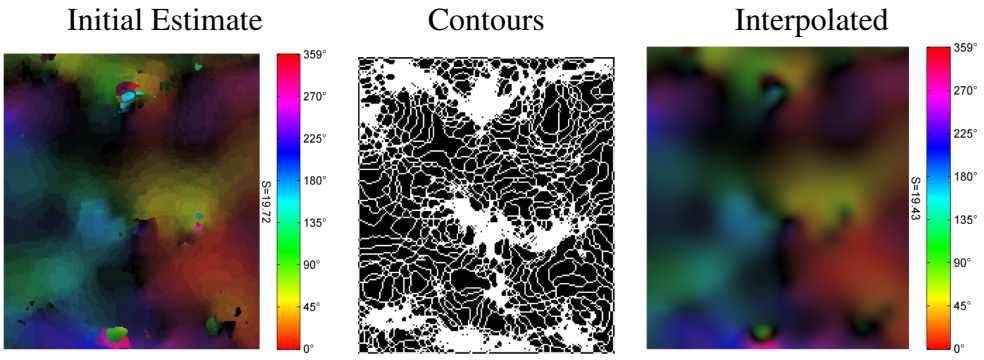


Figure 2.8: The effect of running the contour interpolation algorithm with $\sigma_{\text{smooth}} = 0.2$. Notice that in the areas with artifacts in the original estimate, the contour mask is true throughout.

Possibly the most glaring issue with the simple LSM approach is the lack of non-integer displacement estimates, with sharp transitions between areas of different integer displacements clearly visible as 'plateaus' of constant colour in any of the LSM estimate visualisations. While enlarging the source and target images and then downscaling the resulting displacement field would result in sub-pixel resolution, due to the running time complexity of the LSM algorithm[8] this solution does not scale well. A different approach, used by ELSM, is to interpolate between the integer-valued displacement estimates by fixing the values along the plateau contours and fitting a smooth three-dimensional surface for each of the two displacement components, horizontal and vertical.

This step introduces a new parameter which characterises the smoothness of the fit σ_{smooth}.

2.3.3 Multiscale ELSM (MS-ELSM)

The ELSM approach suffers from another, more subtle issue than the ones solved in the previous subsections, one which is not obvious from the colour-coded visualisations

[8] The complexity of the LSM algorithm is $O(w \times h \times \delta_p \times \delta_s)$, where w and h are the width and height of the images respectively. If the dimensions of the images are doubled then δ_p and δ_s also need to be doubled to cover the same space resulting in a 16-fold increase in running time.

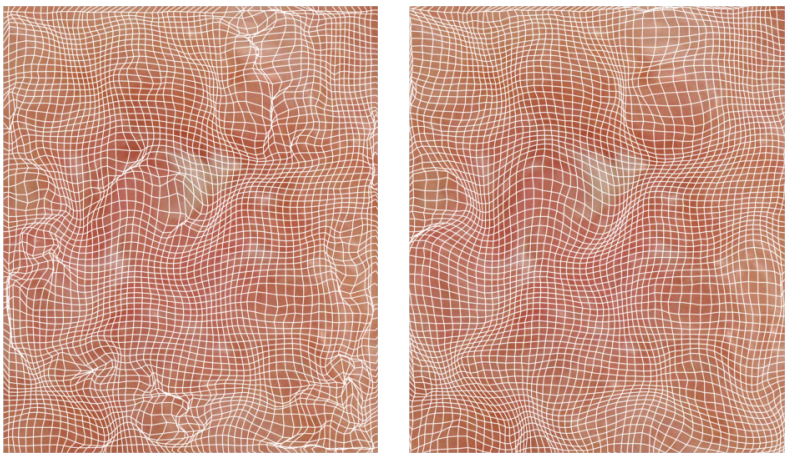


Figure 2.9: Left: An ELSM estimate, showing self-intersections (folds) in the displacement field (such as the one in the top right). Right: The MS-ELSM estimate with the folds corrected.

of displacement fields. Even after interpolating the integer values, the displacement field estimate can be self-intersecting, containing overlapping displacement vectors. Intuitively, two pixels cannot swap positions or 'cross paths' but the model used by the ELSM algorithm does not prevent that. We shall use the term diffeomorphism from category theory to refer to a 'smooth' mapping which does not contain such 'folds'.

2.3.3.1 Sparse Diffeomorphisms

As a simple workaround which ensures the estimated displacement is a diffeomorphism, we could estimate a sparser lattice of displacement vectors and limit their maximal norm to half the distance between two adjacent points in the lattice by setting the search window size, δ_s, to an appropriate value. Then, by interpolating between the vectors in the sparse lattice, each pixel can be assigned a displacement vector and the resulting field would correspond to a diffeomorphism between the two images.

The problem with this workaround is evidently that the resolution would be limited to the resolution of the lattice: a dense lattice results in correct registration of fine detail, but cannot handle large scale deformation (due to the limit on the maximal norm), while a very sparse lattice can model large scale deformation but cannot deal with small features.

2.3.3.2 Multiscale Registration

Despite this problem, the sparse lattice idea does suggest a different approach, partly inspired by [4], which has the benefits of both a sparse and a dense lattice. The idea is to estimate finer detail progressively, rather than attempting to simultaneously estimate both large and small scale deformation. The method relies on building a scale pyramid for each of the images by repeatedly downscaling them to half their size N_{level} times,

such that at level i of the pyramid each image is downscaled by a factor or 2^i compared to the original.

By running the ELSM algorithm repeatedly with a small search window starting with the lowest resolution level in the pyramids, MS-ELSM registers the largest deformations first[9]. The displacement estimated at the lowest-resolution level, N_{level}, can then be scaled up and applied to the source image at the next level of the pyramid, correcting the largest deformations from it; ELSM can then be applied again. This process (the results of which are shown in Figure 2.10) can then be in turn applied to all the levels in the pyramids and the final displacement field estimate will be the sum of all of the partial estimates.

This multiscale technique has another benefit over the basic ELSM approach in addition to guaranteeing that the estimated displacement fields are diffeomorphisms. Since the registration on the low-resolution images essentially performs least squares matching on groups of pixels, the method captures dependencies between adjacent pixels, further alleviating the issue for which the distance penalty was introduced in §2.3.2.3 without requiring the assumption that smaller deformations are more likely.

For more precise fractional displacements, upscaled versions of the source and target images can be added to the pyramid as negative levels, but due to aforementioned problem with ELSM's running time complexity this quickly becomes intractable, so in our tests and experiments we used at most one additional upscaled level, with the images on level -1 at twice the size of the originals.

2.3.4 Probabilistic Multiscale Registration (PMR)

MS-ELSM still relies on the contour interpolation (§2.3.2.4) method to obtain values in areas where the displacement values differ only in fractional parts. This is merely a heuristic and in low-contrast regions, where matching is uncertain, displacement vectors can vary strongly over a short distance due to mismatches. In these areas contour interpolation weighs all the estimates equally. Ideally, in areas of low contrast, only the borders should be taken into account ignoring the mismatches within.

Another issue with MS-ELSM is that, while it removes the need to tune the search window and pixel neighbourhood sizes (δ_s and δ_p, respectively), it still relies on two arbitrary parameters that need to be tuned using a validation set: c_{dist}, the cost incurred by larger scale deformations and σ_{smooth} which controls the 'smoothness' of the contour interpolation.

2.3.4.1 Probabilistic LSM

From a Bayesian point of view, part of the problem is that we estimate a maximum likelihood solution for the displacement field rather than taking the Bayesian approach

[9]This is because a 1-pixel displacement in an image downscaled to half its size n times will, of course, correspond to a 2^n pixel displacement between the original images.

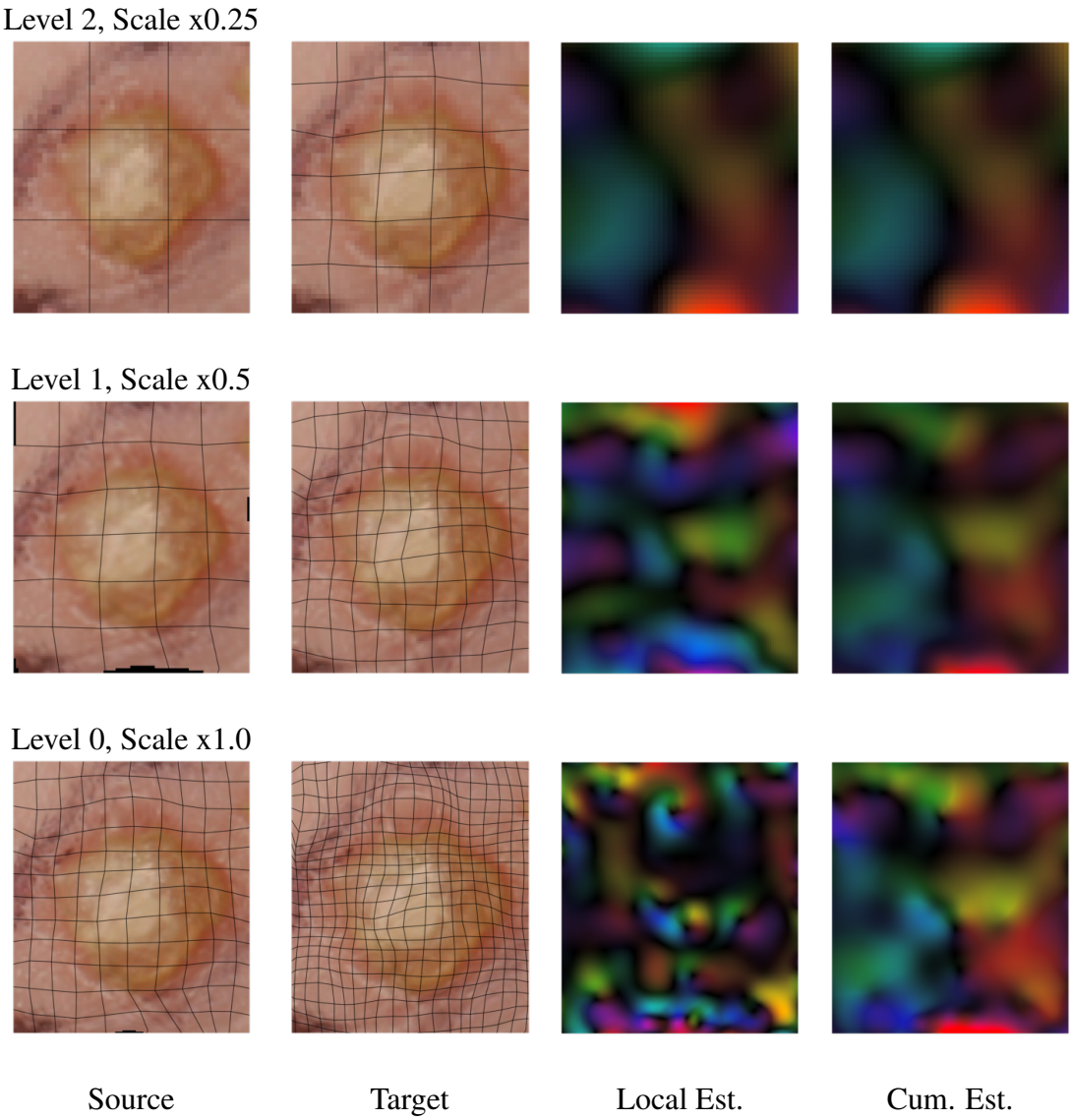


Figure 2.10: An example of Multiscale ELSM with three levels shown in the order in which they are estimated, starting at the lowest resolution level – a fourth of the original size. The *Source* column shows the source image at the current level, with the cumulated flow applied to it (the black grid shows the deformation applied so far); the *Target* column shows the target image at the current level – simply a scaled down version of the original target – and the black grid shows the correct displacement at the current level; the *Local Est.* column shows the displacement field estimated by running ELSM on the 'Source' and 'Target' images of the current level; finally, the *Cum. Est.* column shows the cumulated sum of all the previously estimated displacement fields.

and estimating a distribution to capture the uncertainty of each match. Hence the first step towards implementing a better smoothing method is to replace the core LSM algorithm to one which estimate a distribution over displacement vectors for each pixel rather than simply computing the minimum error match.

Let's assume that adjacent displacement vectors are still independent for now. Esti-

mating a distribution over each vector is relatively straightforward: the least squares approach used by the LSM algorithm has a natural probabilistic interpretation – the log of the density of a joint distribution of independent normally distributed random variables. We shall need to introduce more notation before we can naturally express this distribution: for a given pixel neighbourhood size δ_p and an image A, let a *neighbourhood vector* $\mathrm{nv}_A(\mathbf{x})$ be the $(2\delta_p+1)\times(2\delta_p+1)$ pixel neighbourhood centered at coordinates x in image A, with columns and colour components concatenated into a single column vector of size $3(2\delta_p+1)^2 = 12\delta_p(1+\delta_p)+3$.[10]

Returning to the distribution implied by the LSM algorithm, the implicit assumption is that:

$$\Pr(\mathrm{nv}_T(\mathbf{x}) \mid D_{S\to T}(\mathbf{x}) = \mathbf{d}, S) \sim \mathrm{nv}_S(\mathbf{x}-\mathbf{d}) + \mathcal{N}(0, \sigma^2_{\text{pixel}} I) \tag{2.22}$$

If we also assume a uniform prior for each displacement vector allowed by the search window ($\Pr(D_{S\to T}(\mathbf{x}) = \mathbf{d}) = C_1$, constant) the log posterior for the displacement field becomes equal to a linear function of the LSM *ssd* matching error:

$$\begin{aligned}
\mathcal{L}(\mathbf{x},\mathbf{d}) &= \log\Pr(D_{S\to T}(\mathbf{x}) = \mathbf{d} \mid \mathrm{nv}_T(\mathbf{x}), S) = && (2.23)\\
&= \log\Pr(\mathrm{nv}_T(\mathbf{x}) \mid D_{S\to T}(\mathbf{x}) = \mathbf{d}, S) + \log\Pr(D_{S\to T}(\mathbf{x}) = \mathbf{d}) + C_2 && (2.24)\\
&= -\frac{1}{2\sigma^2_{\text{pixel}}} ssd_{S,T}(\mathbf{x}-\mathbf{d}, \mathbf{x}) + C_1 + C_2 && (2.25)
\end{aligned}$$

Since constants C_1, C_2 and σ^2_{pixel} are irrelevant when maximising $\mathcal{L}$, the maximum posterior solution is precisely equal to the LSM solution.

The form given by Equation (2.25) suggests a few simple improvements to the posterior function. First, instead of using a uniform prior (equal to C_1), we can use a symmetric normal distribution with mean zero – this captures the fact that large deformations are less likely than smaller ones, playing the exact same role as the distance penalty used by MS-ELSM:

$$\Pr(D_{S\to T}(\mathbf{x})) \sim \mathcal{N}(\mathbf{0}, \sigma^2_{\text{prior}}) \tag{2.26}$$

In fact, the log prior term has *precisely* the same form as the distance penalty (defined in Equation (2.21)):

$$\log\Pr(D_{S\to T}(\mathbf{x}) = \mathbf{d}) = -\frac{1}{2\sigma_{\text{prior}}}\|\mathbf{d}\|^2 = c_{\text{dist}}\|\mathbf{d}\|^2 \tag{2.27}$$

The second improvement is related to the assumption described in Equation (2.22). There is no good reason why the pixel neighbourhood distribution should use a spherical normal distribution; in fact one would expect a higher variance in pixels further away from the center of the neighbourhood. So instead, we introduce a diagonal covariance matrix ψ_{pixel} which can be trained from data. Figure 2.11b shows as an example trained from the synthetic dataset.

[10] The 3 comes from the fact that each pixel is in fact a $[0,1]^3$ red-green-blue triplet.

These two changes give rise to the following, new log-posterior function:

$$\mathcal{L}(\mathbf{x},\mathbf{d}) = \mathcal{L}_{\text{likelihood}}(\mathbf{x},\mathbf{d}) + \mathcal{L}_{\text{prior}}(\mathbf{d}) + \log Z \tag{2.28}$$

Where:

$$\mathcal{L}_{\text{prior}}(\mathbf{d}) = -\frac{1}{2\sigma_{\text{prior}}}\|\mathbf{d}\|^2 \tag{2.29}$$

$$\mathcal{L}_{\text{likelihood}}(\mathbf{x},\mathbf{d}) = -\left\|\psi^{-1}_{\text{pixel}}(\text{nv}_T(\mathbf{x}) - \text{nv}_S(\mathbf{x}-\mathbf{d}))\right\|^2 \tag{2.30}$$

If, for a given target pixel at coordinates $\mathbf{x}$, we evaluate this new posterior function for each displacement vector allowed by the search window size and normalise by the sum over all of them we obtain a discretised version of the distribution over the displacement vector at $\mathbf{x}$.

As a simplification, instead of storing these discrete versions directly for each pixel of the target image, the discretised values are used to fit a normal distribution over the displacement vector at $\mathbf{x}$:

$$\Pr(D_{S\to T}(\mathbf{x}) \mid S,T) \sim \mathcal{N}(\mu_{S\to T}(\mathbf{x}), \Sigma_{S\to T}(\mathbf{x})) \tag{2.31}$$

Where $\mu_{S\to T}$ and $\Sigma_{S\to T}$ are functions which return a mean vector and covariance matrix for each integer position in the displacement field. This requires storing only a two-dimensional mean and the three distinct elements of the 2×2 covariance matrix for each displacement vector, resulting in a $w\times h\times 5$ matrix.

A second $w\times h\times 5$ matrix is estimated by estimating the distribution over the inverse displacement field $\Pr(D_{T\to S}(\mathbf{x}) \mid S,T)$, with associated $\mu_{T\to S}$ and $\Sigma_{T\to S}$ similar to the way the bidirectional search works for ELSM.

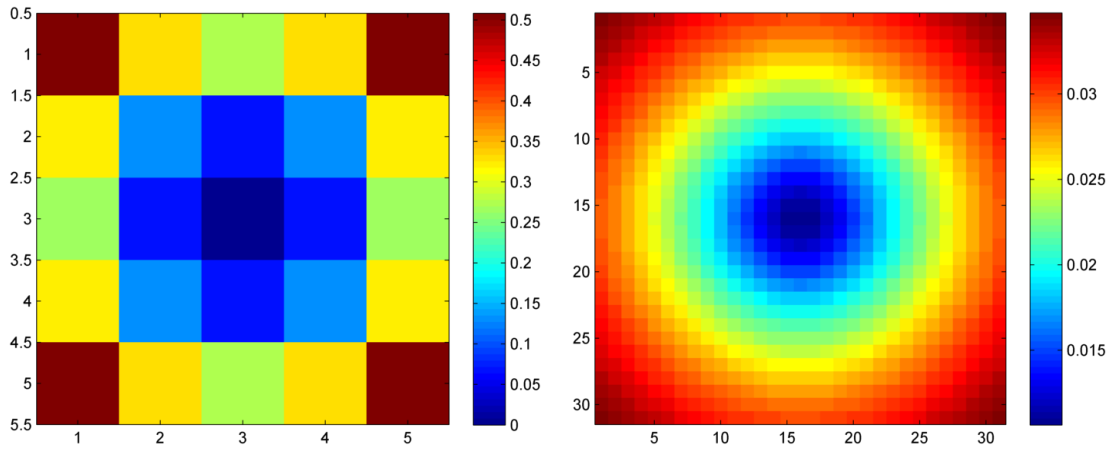
2.3.4.2 Covariance Weighted Smoothing (CWS)

The entire purpose of this redesign was to replace contour interpolation with a method which takes into account uncertainty when the estimating displacement vectors. To this end, we could express the smoothing constraint and the distributions estimated in the previous step as a latent variable model and apply a well-known inference method such as loopy belief propagation to maximise the posterior probability. Instead, we explored a less sophisticated solution, a fast heuristic which performs well empirically.

CWS starts with a displacement field initialised with either $\mu_{S\to T}(\mathbf{x})$ or $\mu_{T\to S}(\mathbf{x})$ at every position, choosing the one with lower variance (more certain match). Then it iteratively smoothes this field by replacing each vector at position $\mathbf{x}$ with a weighted mean of its neighbours and its current value at each step. Let D_i be the estimated field at iteration i, then the iteration step is given by the following relation:

$$D_{i+1}(\mathbf{x}) = Z^{-1}\left(\left(\Sigma^{-1}_{S\to T}(\mathbf{x}) + \Sigma^{-1}_{T\to S}(\mathbf{x})\right)D_i(\mathbf{x}) + \sum_{\mathbf{d}\in Nh(\delta_{\text{cws}})}\frac{1}{2\omega(\mathbf{d})}D_i(\mathbf{x}+\mathbf{d})\right) \tag{2.32}$$

$$Z = \Sigma^{-1}_{S\to T}(\mathbf{x}) + \Sigma^{-1}_{T\to S}(\mathbf{x}) + \sum_{\mathbf{d}\in Nh(\delta_{\text{cws}})}\frac{1}{2\omega(\mathbf{d})} \tag{2.33}$$



(a) (b)

Figure 2.11: (a) the variance between a displacement vector and its neighbours, estimated by ω, with $\delta_{\text{cws}} = 2$. The variance is zero in the centre, of course and it increases with distance from the center as expected; (b) the Ψ_{pixel} covariance estimated from a synthetic dataset, with $\delta_p = 15$.

In the expression for $D_{i+1}(\mathbf{x})$ there are two terms. The first one is the current value of the field $D_{i+1}(\mathbf{x})$ weighted by the sum of the 'precision matrices' (inverse covariance matrices) of the two fitted distributions (forward and inverse). The second term is a weighted sum of the vectors neighbours in the current estimate of field, with weights given by $\omega(\mathbf{d})$. Z is simply the sum of all the weights. Inuitively, this method works by aligning the vectors with their neighbours at a speed based on the estimated distributions' variance along the direction required for alignment. A vector whose distribution has low variance will 'resist change' more, forcing its neighbours to align to its value rather than the other way around.

The values of $\omega(\mathbf{d})$ can be trained from a set of displacement fields $D^{(n)}_{S\to T}(x), \forall n \in 1\ldots N$:

$$\omega(\mathbf{d}) = \frac{1}{N}\sum_{n=1}^{N} \|D^{(n)}_{S\to T}(x) - D^{(n)}_{S\to T}(x+d)\|^2 \tag{2.34}$$

An example values of ω trained on a synthetic dataset can be seen in Figure 2.11a.

2.3.5 FastPD-Optimised Markov Random Fields (MRF-FastPD)

The last algorithm we present is an established DDR method introduced by [4] which models deformation as a cubic B-spline, with the positions of the spline's control points as random variables within a Markov Random Field (MRF). The MRF energy is defined as a sum of two terms E_{data} and E_{smooth}, with the former modelling the fitness of a setting of B-spline control points based on the difference between the transformed source image and the target and the latter a function of the displacement field gradients which penalises large gradients to favour smooth fields.

To generate diffeomorphisms, a similar technique to the one presented in §2.3.3 is used: the B-spline control point density is progressively increased, while at each step a hard constraint on the maximal displacement for each control point is set to 0.4 of the distance between adjacent points.

The MRF optimisation is performed using a high performance, linear programming method called Fast-PD.

2.4 Measuring Changes in Pigmentation

The method for measuring pigmentation changes is very simple – first, the source image is distorted using the displacement field estimated in the previous step, so that persistent features are in the same position in both images. Then, by subtracting the RGB values of one image from the other we obtain a three channel map corresponding to the source to target (potentially negative) RGB differences, aligned with the target image.

Chapter 3

Experiments

3.1 Experimental Procedure

The image alignment and deformable registration methods were evaluated separately. First, to compare the four DDR algorithms three synthetic test sets are used, of increasing deformation levels, to enable comparison against the ground-truth displacement field (see subsection §3.2.1 for details of how the synthetic data is generated). The images are not transformed in any other way, to obtain results independent of the effectiveness of the alignment algorithm.

To evaluate the image alignment algorithm an affine transformation was first applied to the target images from the synthetic dataset. Then the algorithm was ran on the source image and the transformed target image and the estimated homography compared to the ground truth.

3.1.1 Data

The data gathered specifically for this project were photographs of six different lesions from a single volunteer taken on two occasions 38 days apart. Each lesion was photographed using a pair of consumer digital cameras and a ring flash attached to a rigid metal frame to ensure a fixed distance and relative angle to the photographed patch of skin. The pictures from the two different cameras could be used to estimate depth data, but this was not required for this project.

In addition to these twenty-four (six lesions, two cameras, two occasions) images, we also had access to a larger set of 960 single images of lesions of five different different types[1], with no follow-up images. These were used to create the synthetic dataset described in §.

[1] Actinic keratosis, basal cell carcinoma, malignant melanoma, squamous cell carcinoma and seborrheic keratoses.

3.2 Dense Deformable Registration

3.2.1 Synthetic Data

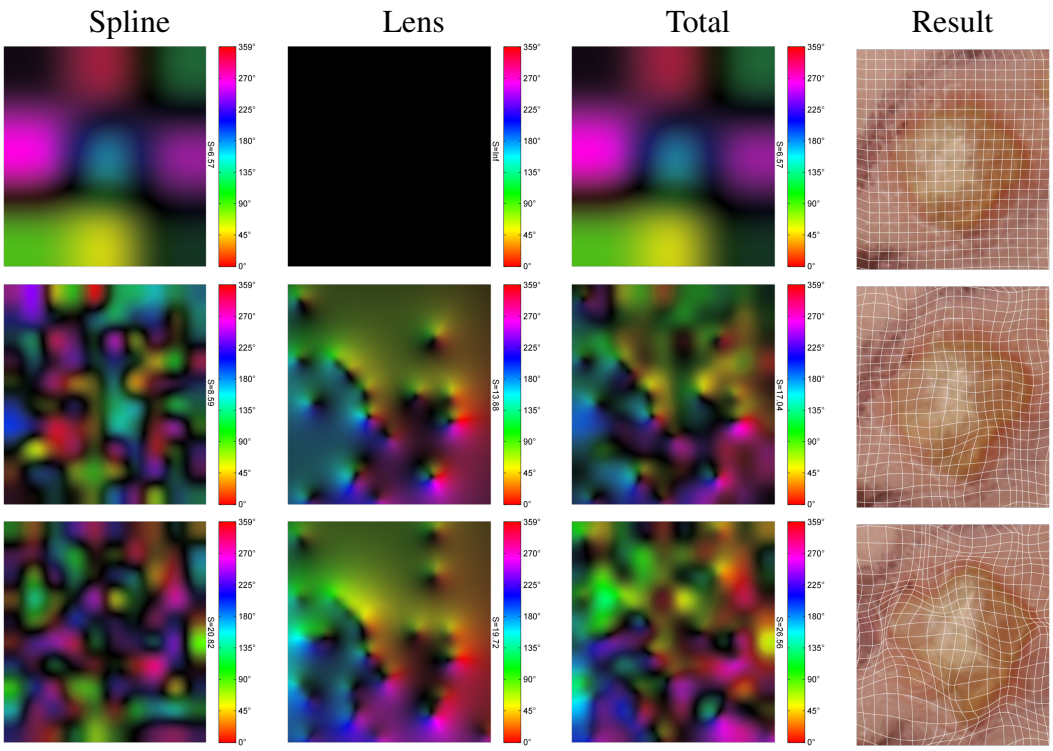


Figure 3.1: Left to Right: the displacement fields from each deformation method, their sum and the resulting target image. Top to Bottom: the parameters from the three different datasets EASY, MEDIUM, HARD. Note that the displacement fields are on vastly different scales (the $S = \ldots$ on the right).

To evaluate the deformable registration algorithms we require a ground truth displacement field for comparison. Out of the 960 single lesions we randomly selected 300 images for a test set and 90 for training. These images are used as sources and deformed them artificially with different parameters to obtain a synthetic target image for each.

The two deformation methods described below are employed to generate two displacement fields which are added to obtain the final ground truth displacement field:

1. **Bicubic Spline Deformation**. This is the same method used by the authors of [4] to generate the dataset used to evaluate MRF-FastPD. It works by generating a sparse lattice of random initial displacement vectors and then bicubically interpolating between them to obtain a vector for each pixel. The initial vectors are drawn from a normal distribution with mean zero and variance σ^2_{spline} and their maximal norm capped at the distance between adjacent vectors (this ensures the field is a diffeomorphism, see §2.3.3 for an explanation). The density of the initial lattice is given by a second parameter ρ_{spline} equal to the number of initial

vectors per pixel – for instance a $\rho_{\text{spline}} = 0.05$ results in one vector every 20 pixels of the source image.

2. **Lens Deformation**. The spline deformation generates a very 'uniform', constant frequency displacement field (see Figure 3.1), so an additional pass over the displacement field is performed. A number N_{lens} of 'lenses' are added at random positions to the displacement field. Pixels are pushed away from the center with a strength which decays polynomially with distance. A single lens centered at position $\mathbf{c}$ adds a displacement field described by:

$$\text{LensField}_{S\to T}(\mathbf{x}) = (\mathbf{x}-\mathbf{c})\frac{s_{\text{lens}}}{(\|\mathbf{x}-\mathbf{c}\|+\varepsilon_{\text{lens}})^{d_{\text{lens}}}} \tag{3.1}$$

Where s_{lens} is the 'strength' of the field, d_{lens} the exponent of the polynomial decay and $\varepsilon_{\text{lens}}$ a constant which offsets the distance to avoid very large displacements at small distances. These three scalars, together with the number of lenses N_{lens} are parameters which control the shape and magnitude of the generated lens deformation.

As mentioned before, these two deformation methods are employed to generate an artificial displacement field which is applied to the source image to obtain an initial synthetic target image. Finally, independent Gaussian noise with variance σ^2_{noise} is added to each colour component of each pixel to obtain the final synthetic target image.

Parameter	EASY	MEDIUM	HARD	Description
σ^2_{spline}	7.84	7.84	36	Variance of initial spline vectors.
ρ_{spline}	0.01	0.03	0.03	Spline density.
N_{lens}	0	20	30	Number of lenses.
s_{lens}	-	2.8	2.9	Strength of lens fields.
$\varepsilon_{\text{lens}}$	-	0.14	0.06	Lens field distance offset.
d_{lens}	-	0.75	0.75	Lens polynomial decay exponent.
σ^2_{noise}	1/1600	1/400	1/100	Variance of pixel noise.

Table 3.1: The parameter values used for the three test sets.

Three different test sets (labelled EASY, MEDIUM and HARD) were created using the same set of 100 source images, but with different deformation parameters. Table 3.1 gives the parameter values used for each of the three datasets.

Additionally, a training set of 90 source-target image pairs was created (using a different set of source images, of course) with a third of the target images generated using the EASY deformation parameters, a third using the MEDIUM ones and the last third using the HARD ones. Note that all the algorithms are trained only once on this training set and then tested on each of the different test sets. They are not trained and tested separately for each setting of the parameters.

3.2.2 Parameter Tuning

3.2.2.1 ELSM

Step	δ_s	δ_p	c_{dist}	σ_{smooth}
1	9	6	0	0
2	9	6	1.3136×10^{-5}	0
3	9	6	1.3136×10^{-5}	0.25
4	11	4	1.3136×10^{-5}	0.25

Table 3.2: The ELSM parameter estimates after each step of the tuning algorithm.

The ELSM method has the most parameters that need to be tuned: δ_p (the pixel neighbourhood size), δ_s (the search window size), c_{dist} (the distance penalty) and σ_{smooth} (the smoothing parameter). Due to time constraints, only 10% of the training dataset was used to tune the ELSM parameters. The algorithm used for parameter tuning works as follows:

1. Set c_{dist} and σ_{smooth} to 0 and optimise the neighbourhood and search window sizes by minimising RRMSE over all values $\delta_p \in \{2 \dots 12\}$ and $\delta_s \in \{5, 7, 9, \dots, 17\}$.
2. Using the δ-s estimated in step 1 optimise c_{dist} independently.
3. Optimise σ_{smooth} independently, using the δ-s estimated in step 1 and the c_{dist} estimated in step 2.
4. Finally, re-estimate the pixel neighbourhood and search window sizes using the values of c_{dist} and σ_{smooth} estimated in steps 2 and 3, respectively.

Table 3.2 shows the parameter estimates after each step of this algorithm, while Figure 3.2 shows the error functions being minimised at each step.

3.2.2.2 MS-ELSM

Step	δ_p	c_{dist}	σ_{smooth}
1	5	0	0
2	5	2.8571×10^{-6}	0
3	5	2.8571×10^{-6}	0.2
4	3	2.8571×10^{-6}	0.2

Table 3.3: The MS-ELSM parameter estimates after each step of the tuning algorithm.

The multiscale ELSM algorithm does not require search window size tuning, resulting in one fewer parameter to tune. Using the same tuning algorithm as in the previous subsection (but optimising just δ_p in steps 1 and 4) we obtain the values from Table 3.3.

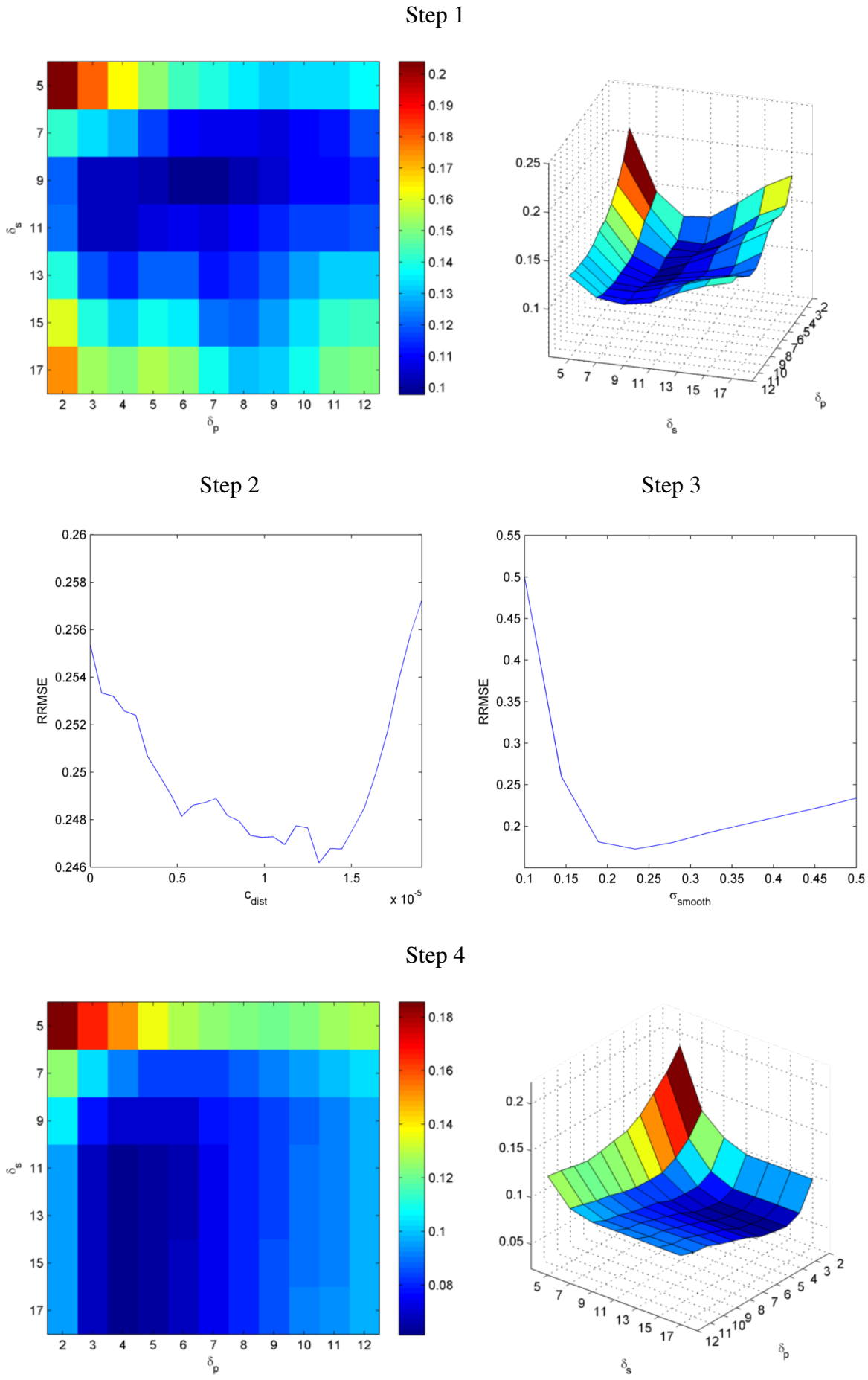


Figure 3.2: The error functions being optimised at each step of the ELSM tuning algorithm. Steps 1 and 4 show RRMSE w.r.t. δ_s (y-axis) and δ_p (x-axis); steps 2 and 3 show RRMSE w.r.t. to c_{dist} and σ_{smooth} respectively.

3.2.2.3 PMR

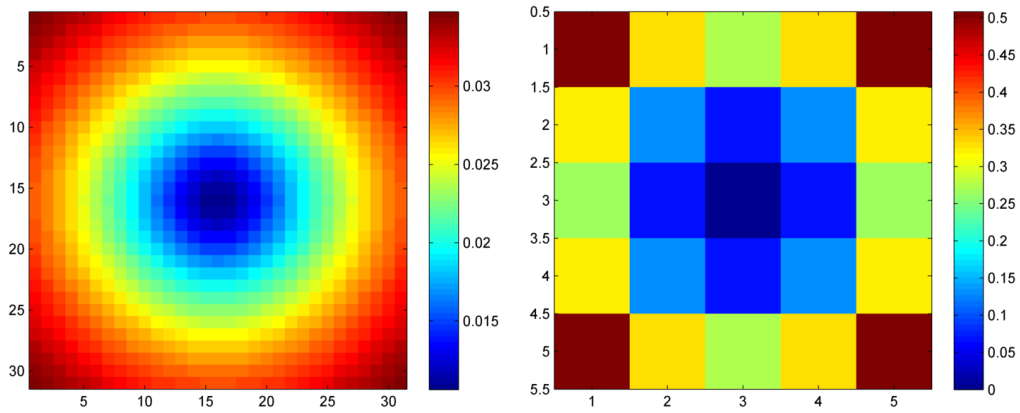


Figure 3.3: The variances stored in the diagonal covariance matrices ψ_{pixel} (left) and ω (right), arranged such that they correspond to the position relative to the centre pixel. It can be seen in both covariances that, as expected, the variances get higher as one gets further away from the matched pixel.

PMR relies on the whole training set to estimate the ψ_{pixel} and ω diagonal covariance matrices (by measuring the variance between neighbouring displacement vectors).

The pixel neighbourhood covariance, ψ_{pixel}, is estimated by as sample mean of the squared differences between pixel neighbourhoods matched by the ground truth training displacement fields, while the CWS neighbour weighting, ω, is estimated by the sample mean of squared differences between each displacement vector and its neighbours over all of the the training displacement fields (the training process is described in §2.3.4.2).

The two estimates obtained from the training set are shown in Figure 3.3.

3.2.3 Results

The four different algorithms (ELSM, MS-ELSM, MR-FastPD and PMR) were ran on the three test sets (EASY, MEDIUM and HARD) and four metrics were considered:

1. Displacement Root Mean Square Error (DispErr). Measures the deviation of the estimated displacement field ($\tilde{D}_{S\to T}$) from the ground truth displacement ($D_{S\to T}$) using an RMSE measure. Let $\Omega = \{1\dots w\} \times \{1\dots h\}$, then:

$$\text{DispErr}(\tilde{D}_{S\to T}) = \sqrt{\frac{1}{wh}\sum_{\mathbf{x}\in\Omega}\left\|\tilde{D}_{S\to T}(\mathbf{x}) - D_{S\to T}(\mathbf{x})\right\|^2} \tag{3.2}$$

2. Image Root Mean Square Error (ImgErr). Measures the deviation of the application[2] of the estimated field to the source image from the ground truth application

[2] Application in the sense described in §2.3.1.2 – deforming an image using a displacement field.

Metric	Dataset	MRF-FastPD	ELSM	MS-ELSM	PMR
DispRelErr	EASY	$0.0234 \pm 0.008^*$	0.1367 ± 0.052	0.1405 ± 0.054	0.0427 ± 0.016
	MEDIUM	0.0428 ± 0.009	0.1422 ± 0.032	0.0726 ± 0.028	$0.0343 \pm 0.006^*$
	HARD	$0.1242 \pm 0.077^*$	0.5023 ± 0.084	0.3323 ± 0.448	$0.1301 \pm 0.102^*$
ImgRelErr	EASY	$0.0053 \pm 0.005^*$	0.0302 ± 0.014	0.0298 ± 0.012	0.0102 ± 0.005
	MEDIUM	$0.0119 \pm 0.014^*$	0.0365 ± 0.021	0.0176 ± 0.018	$0.0110 \pm 0.010^*$
	HARD	$0.0354 \pm 0.024^*$	0.1500 ± 0.050	0.0954 ± 0.139	$0.0415 \pm 0.029^*$
DispErr	EASY	$0.0440 \pm 0.006^*$	0.2549 ± 0.042	0.2611 ± 0.031	0.0790 ± 0.004
	MEDIUM	0.1156 ± 0.019	0.3909 ± 0.102	0.1953 ± 0.067	$0.0927 \pm 0.013^*$
	HARD	$0.7372 \pm 0.505^*$	2.9494 ± 0.821	1.9276 ± 2.533	$0.7683 \pm 0.627^*$
ImgErr	EASY	$0.0009 \pm 0.001^*$	0.0052 ± 0.002	0.0051 ± 0.002	0.0018 ± 0.001
	MEDIUM	$0.0020 \pm 0.002^*$	0.0063 ± 0.004	0.0030 ± 0.003	$0.0019 \pm 0.002^*$
	HARD	$0.0061 \pm 0.004^*$	0.0261 ± 0.010	0.0160 ± 0.022	$0.0072 \pm 0.005^*$

Table 3.4: The results of the four DDR methods on each of the three datasets, showing the mean $\pm$ std. dev. for the four different metrics over each of the datasets. The asterisks mark the best performing method from that row (if multiple methods have an asterisk the difference between them is not statistically significant).

– the image obtained by applying the ground truth field to the source image.[3] Analogously to DispErr:

$$\mathrm{ImgErr}(\tilde{D}_{S\to T}) = \sqrt{\frac{1}{wh}\sum_{\mathbf{x}\in\Omega}\left\|S(\mathbf{x}-\tilde{D}_{S\to T}(\mathbf{x})) - S(\mathbf{x}-D_{S\to T}(\mathbf{x}))\right\|^2} \tag{3.3}$$

3. Displacement Relative RMSE (DispRelErr). Normalises DispErr by the standard deviation of the field, to address the fact that displacement fields with low variation are easier to estimate.

4. Image Relative RMSE (ImgRelErr). Defined analogously to DisplRelErr, but working on the field applications rather than on the fields themselves.

The results of the described experiments are shown in Table 3.4. The relative errors are easier interpret; for example, according to the DispRelErr metric, while the MS-ELSM estimates are on average wrong by ≈ 0.14 standard deviations from the ground truth on the EASY dataset, the MRF-FastPD only deviate by $\approx 0.02\sigma$, a seven-fold improvement. A few facts can be inferred from the results:

- As expected, MRF-FastPD and PMR consistently outperform the other two methods on all of the datasets by any metric.
- ELSM and MS-ELSM perform similarly on the EASY dataset, while MS-ELSM performs significantly better than ELSM on the MEDIUM dataset.
- Under the displacement metrics PMR performs signifcantly better than MRF-FastPD on the MEDIUM dataset, worse on the EASY and indistinguishably on the HARD dataset.

[3] The ground truth application differs from the target because of noise.

- The image metrics are not as good as distinguishing between the methods as the displacement metrics.

3.3 Image Alignment

	EASY	MEDIUM	HARD
Mean $\pm$ Std	0.1413 ± 0.1122	0.2216 ± 0.2364	0.5644 ± 0.6623

Table 3.5: The image alignment evaluation results.

To evaluate the performance of the image alignment algorithm each target image from the DDR testing sets (§3.2.1) was transformed with a random affine transformation, with a maximum rotation of $\pm 10^{\circ}$, maximum asymmetric scaling (independent on each axis) of $\pm 5\%$ and maximum translation of $\pm 5\%$ of the image size. The reported error is computed as the mean element-wise deviation between the estimated and ground truth homography matrices, normalised by the standard deviation of each element over the ground-truth matrices. The results are shown in Table 3.5.

Chapter 4

Conclusions and Future Work

This project set out to explore approaches for quantifying skin lesion changes over time and, while it achieved most of its goals, a deeper, more ambitious exploration of this almost wholly unresearched subject would require more resources and time than those which were available.

A major problem which we were not able to address was the almost complete lack of **real data**, as there were only six follow-up images of real patients' lesions. Not only did this impair evaluation, but it also affected the research process itself, as the relative importance of different issues (such as alignment versus pigmentation versus deformation) is not clear in the absence of real data. It is our belief that any further work in this area will require a large dataset of image pairs, potentially with gold standard labels. The latter would be particularly important given the potential for the information about lesion change in augmenting existing automated skin lesion classifiers such as the one presented in [2].

This project focused more on introducing and evaluating the PMR deformable registration method rather than attempting to compare a **large number of DDR methods**. With access to real data it would be valuable to see which DDR methods are best applied to this problem. Furthermore, the current form of the PMR algorithm leaves much room for improvement. We will only mention two obvious directions which we did not explore due to time constraints. First, the algorithm is very slow as its running time complexity scales very badly; in its current form, even though it is written in highly-optimised multi-threaded C++, it is still around an order of magnitude slower than MRF-FastPD on a 16 core CPU. An area worth exploring to alleviate this issue is the pixel matching distribution estimation (the slowest part of the algorithm). Instead of evaluating all of the matches within the search window to fit the distributions, one could use only a well picked subset.

The second issue is potentially the reason the algorithm disproportionately underperforms on the EASY dataset. The posterior maximisation step of the **PMR method** (Covariance Weighted Smoothing, §2.3.4.2) is very much a heuristic and more principled methods should be attempted. Anything from loopy belief propagation to simulated annealing could potentially improve the overall performance. Even with this method,

the algorithm could have a better halting condition than the fixed number of iterations currently used. For example, by evaluating the ImgErr metric after each step, stopping if the error has been consistently increasing for more than a number of steps and choosing the field which resulted in the lowest error from all the iteration steps. This is the problem in the EASY dataset, the smoothing stops after too few iterations, and the field ends up being too 'wavy'. See Figure 4.1.

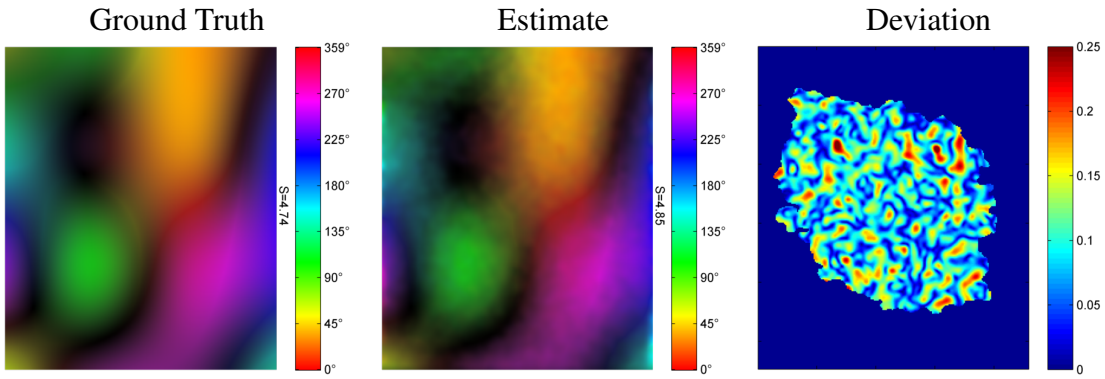


Figure 4.1: Due to an insufficient number of iterations, the PMR estimates are too 'wavy' for the EASY dataset.

In terms of the **image alignment** step, more homography estimation methods should be implemented and evaluated, as the SiFT + RANSAC technique is far from perfect, particularly since the two matched images are not identical (remember that the target image is non-linearly deformed). An interesting idea for improving alignment regardless of the method used is to make use of the DDR information: rather than assuming that the images are perfectly aligned when performing DDR, a two-step process could be used. First, the images are aligned and DDR. Then the estimated displacement vectors can be used to estimate a second affine transformation (after all, a displacement field encodes matches between the source and target images). This second transformation can be applied to the source image and DDR performed again to obtain the final displacement field.

From an **evaluation** point of view, there are a number of smaller issues which we were not able to address during this project: the MRF-FastPD parameters could not be tuned automatically so the default values were used, a larger dataset should have been used to be able to discriminate between the MRF-FastPD and PMR performance on the HARD dataset and more metrics could be used for comparing DDR methods such as the ones used in [4].

Bibliography

[1] Murace R Dragonetti E Manganaro M Guerra O Bizzi S Baldi A, Quartulli M. Automated dermoscopy image analysis of pigmented skin lesions. *Cancers*, (2):262–274, 2010.

[2] Lucia Ballerini, Robert B Fisher, Ben Aldridge, and Jonathan Rees. A color and texture based hierarchical k-nn approach to the classification of non-melanoma skin lesions. In *Color Medical Image Analysis*, pages 63–86. Springer, 2013.

[3] Pierre Castadot, John Aldo Lee, Adriane Parraga, Xavier Geets, Benoît Macq, and Vincent Grégoire. Comparison of 12 deformable registration strategies in adaptive radiation therapy for the treatment of head and neck tumors. *Radiotherapy and oncology*, 89(1):1–12, 2008.

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

[5] M.B. Lens and M. Dawes. Global perspectives of contemporary epidemiological trends of cutaneous malignant melanoma. *British Journal of Dermatology*, 150(2):179–185, 2004.

[6] Xiang Li, Ben Aldridge, Lucia Ballerini, Robert Fisher, and Jonathan Rees. Depth data improves skin lesion segmentation. In Guang-Zhong Yang, David Hawkes, Daniel Rueckert, Alison Noble, and Chris Taylor, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2009*, volume 5762 of *Lecture Notes in Computer Science*, pages 1100–1107. Springer Berlin Heidelberg, 2009.

[7] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2.

[8] I. Maglogiannis and C.N. Doukas. Overview of advanced computer vision systems for skin lesions characterization. *Information Technology in Biomedicine, IEEE Transactions on*, 13(5):721–733, 2009.

[9] Scott W Menzies, Alex Gutenev, Michelle Avramidis, Andrew Batrac, and William H McCarthy. Short-term digital surface microscopic monitoring of atypical or changing melanocytic lesions. *Archives of dermatology*, 137(12):1583, 2001.