Feature-Driven Direct Non-Rigid Image Registration

Florent Brunet* Vincent Gay-Bellile† Adrien Bartoli‡ Nassir Navab§ Rémy Malgouyres¶

Pre-print version of our paper accepted at:
International Journal of Computer Vision, Volume 93, Number 1, 33-52, March 2011

Abstract

The direct registration problem for images of a deforming surface has been well studied. Parametric flexible warps based, for instance, on the Free-Form Deformation or a Radial Basis Function such as the Thin-Plate Spline, are often estimated using additive Gauss-Newton-like algorithms. The recently proposed compositional framework has been shown to be more efficient, but cannot be directly applied to such non-groupwise warps.

Our main contribution in this paper is the Feature-Driven framework. It makes possible the use of compositional algorithms for most parametric warps such as those above mentioned. Two algorithms are proposed to demonstrate the relevance of our Feature-Driven framework: the Feature-Driven Inverse Compositional and the Feature-Driven Learning-based algorithms. As another contribution, a detailed derivation of the Feature-Driven warp parameterization is given for the Thin-Plate Spline and the Free-Form Deformation. We experimentally show that these two types of warps have a similar representational power. Experimental results show that our Feature-Driven registration algorithms are more efficient in terms of computational cost, without loss of accuracy, compared to existing methods.

Keywords. Direct registration inverse compositional image alignment deformable model.

1 Introduction

Registering images of a deforming surface is important for tasks such as video augmentation by texture editing, deformation capture and non-rigid Structure-from-Motion. This is a difficult problem since the appearance of imaged surfaces varies due to several phenomena such as camera pose, surface deformation, lighting and motion blur. Recovering a 3D surface, its deformations and the camera pose from a monocular video sequence is intrinsically ill-posed. While prior information can be used to disambiguate the problem, see *e.g.* (Bregler et al, 2000; Gay-Bellile et al, 2006; Pilet et al, 2005), it is common to avoid a full 3D model by using image-based deformation models, *e.g.* (Bartoli and Zisserman, 2004; Bookstein, 1989; Cootes et al, 2004; Lim and Yang, 2005). The Thin-Plate Spline warps (TPS) is one possible deformation model, proposed in a seminal paper by (Bookstein, 1989), that has been shown to effectively model a wide variety of image deformations in different contexts. Recent work shows that the TPS warp can be estimated not only with the traditional landmark based method, but also with direct methods, *i.e.* by minimizing the intensity discrepancy between registered images (Bartoli and Zisserman, 2004; Lim and Yang, 2005). Other non-rigid warps include Radial Basis Functions (with *e.g.* multiquadrics (Little et al, 1997) or Wendland's (Fornefett et al, 1999) as kernel function) and Free-Form Deformations (Rueckert et al, 1999).

The Gauss-Newton algorithm with additive update of the parameters is usually used for conducting the minimization. Its main drawback is that the Hessian matrix must be recomputed and inverted at each iteration. More efficient solutions have been proposed by (Baker and Matthews, 2004) based on compositional updating of the parameters. They might lead to a constant Hessian matrix. Most non-rigid warps do not form groups, preventing the use of compositional algorithms which require one to compose and possibly invert the warps. Despite several attempts to relax the groupwise assumption by various approximations (Gay-Bellile et al, 2006; Matthews and Baker, 2004; Romdhani and Vetter, 2003), there is no simple solution in the literature.

This paper is an extended version of an earlier conference version (Gay-Bellile et al, 2007). With respect to the literature, it brings several contributions:

- The main contribution of this paper is the *Feature-Driven registration concept*. It allows one to devise compositional algorithms while relaxing the strict groupwise warp requirement, and is thus applicable to most non-rigid parametric warps. The main idea is to parameterize the warp by a set of *driving features* instead of the usual control points or coefficients, and to act on these features directly.
- Two operations, *reversion* and *threading*, are defined using the Feature-Driven parameterization. They respectively approximate inversion and composition when those are not guaranteed to exist or cannot be easily computed.
- Using our Feature-Driven framework, we extend the Inverse Compositional algorithm to non-rigid warps. Our framework also allows us to propose a forward Learning-based algorithm for non-rigid warps. Besides, we improve the classical linear learning algorithm using a new piecewise linear relationship.
- We give a detailed derivation of the Feature-Driven parameterization for the TPS and FFD warps. In particular, we present an extrapolation method for the FFD warp, required

*ISIT / Université d'Auvergne – Clermont-Ferrand, France // CAMPAR – TUM – München, Germany
†CEA, LIST, Vision and Content Engineering Laboratory, Point Courrier 94, Gif-sur-Yvette, F-91191 France
‡ISIT / Université d'Auvergne – Clermont-Ferrand, France
§CAMPAR – TUM – München, Germany
¶LIMOS, UMR 6158 – Clermont-Ferrand, France

for the Feature-Driven framework. We also show that the TPS and FFD warps have a similar representational power.

We experimentally show that Feature-Driven algorithms are clearly more efficient without loss of accuracy compared to previous state-of-the-art methods. The combination of the Feature-Driven framework with Learning-based local registration outperforms other algorithms for most experimental setups.

Roadmap. Previous work is reviewed in §2. In particular, previous attempts to extending compositional algorithms to non-groupwise warps are presented in §2.2. The Feature-Driven framework and the associated operations are explained in §3. Registration with the Feature-Driven Inverse Compositional and the Feature-Driven Learning-based algorithms are described in §4. The Feature-Driven parameterization of the TPS and of the FFD warps are detailed in §5. Experimental results on simulated and real data are reported in §6. Conclusions and further work are discussed in §7. Details on the piecewise linear relationship we used for the Learning-based local registration step are given in appendix A.

Notation. Scalars are in italics (x), vectors in bold ($\mathbf{v}$), matrices in sans-serif (M) and sets (or collections) in fraktur ($\mathfrak{C}$). Vectors are always considered as column vectors. The inverse of a matrix M is written M^{-1}, the pseudo-inverse $\mathsf{M}^{\dagger}$ and the transpose M^{T}. The symbol $\mathbb{R}$ denotes the set of the real numbers. The identity matrix of size n is denoted I_n. The notation $\mathbf{0}_{m\times n}$ and $\mathbf{1}_{m\times n}$ corresponds to the matrices of size $m \times n$ filled with zeros and ones respectively. The operator that vectorizes a matrix is denoted $\boldsymbol{\nu}$, *i.e.* $\boldsymbol{\nu}(\mathsf{M}) = \begin{pmatrix}\mathbf{m}_1^{\mathsf{T}} & \dots & \mathbf{m}_l^{\mathsf{T}}\end{pmatrix}^{\mathsf{T}}$ where the vectors $\{\mathbf{m}_i\}_{i=1}^{l}$ are the columns of M. Conversely, the operator ζ_p builds a matrix of size $\mathbb{R}^{q\times p}$ from a vector of size $\mathbb{R}^{pq}$, *i.e.* $\zeta_p(\mathbf{v}) = (\mathbf{v}_1 \ \dots \ \mathbf{v}_p) \in \mathbb{R}^{q\times p}$ where $\mathbf{v} = \begin{pmatrix}\mathbf{v}_1^{\mathsf{T}} & \dots & \mathbf{v}_l^{\mathsf{T}}\end{pmatrix}^{\mathsf{T}} \in \mathbb{R}^{pq}$. The notation ζ is used to abbreviate ζ_2. We denote $\mathrm{rms}(\mathbf{v})$ the Root Mean of Squares (RMS) of the m-vector $\mathbf{v}$, *i.e.* $\mathrm{rms}(\mathbf{v}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\mathbf{v}_i^2} \propto \|\mathbf{v}\|$, with $\|\bullet\|$ the two-norm.

Images are considered as $\mathbb{R}^2 \to \mathbb{R}$ functions[1] and are denoted using calligraphic fonts ($\mathcal{A}$). If $\mathfrak{C}$ is a collection of pixels then $\boldsymbol{\xi}_{\mathfrak{C}}(\mathcal{I})$ is the vector in which are stacked the values of $\mathcal{I}$ for all the pixels indicated in $\mathfrak{C}$. More precisely, if $\mathfrak{C} = \{\mathbf{q}_i\}_{i=1}^{|\mathfrak{C}|}$ then $\boldsymbol{\xi}_{\mathfrak{C}}(\mathcal{I}) = \begin{pmatrix}\mathcal{I}(\mathbf{q}_1) & \dots & \mathcal{I}(\mathbf{q}_{|\mathfrak{C}|})\end{pmatrix}^{\mathsf{T}} \in \mathbb{R}^{|\mathfrak{C}|}$ where $|\mathfrak{C}|$ is the cardinal of the set $\mathfrak{C}$.

The images to be registered are written $\mathcal{I}_i$ with $i = 1, \dots, n$. The texture image, *e.g.* the region of interest in the first image, is denoted $\mathcal{I}_0$. The set of pixels of interest, *i.e.* the subset of pixels of the image $\mathcal{I}_0$ actually used to estimate a warp, is denoted $\mathfrak{R}$. A generic parametric warp is written $\mathcal{W}$. It depends on a parameter vector $\mathbf{u}_i$ for image $\mathcal{I}_i$ and maps a point $\mathbf{q}_0$ from the texture image to the corresponding point $\mathbf{q}_i$ in the i-th image: $\mathbf{q}_i = \mathcal{W}(\mathbf{q}_0; \mathbf{u}_i)$. The notation $\mathcal{W}(\mathbf{q}; \bullet)$ designates the warp as a function of its parameters, *i.e.* an $\mathbb{R}^l \times \mathbb{R}^2$ function where l is the size of the parameter vector, instead of as a function of the pixels.

[1] In practice, images are $\mathbb{R}^2 \to \mathbb{R}^c$ functions where c is the number of channels. For the sake of simplicity and without loss of generality, we consider that $c = 1$ in all the derivations of this article.

2 Problem Statement and Previous Work

The registration of images of deformable surfaces has received a growing attention over the past decade. Usually, for purely twodimensional registration, as is the case in this paper, smoothness constraints are used to filter out the noise and 'fill-in' the optical flow in untextured image areas. These soft constraints are either implicitly incorporated in a parameterized warp or enforced through regularization. In this paper, we focus on direct as opposed to feature-based methods, *e.g.* (Pilet et al, 2005; Torr and Zisserman, 1999). In the feature-based methods, the warp parameters are estimated from features, such as points, which have first been extracted and matched in the images to register (Szeliski, 2006). Note that there exist methods that mix both the feature-based and the direct approaches to image registration. See for instance, (Johnson and Christensen, 2002; Georgel et al, 2008).

Direct registration consists in minimizing the pixel value discrepancy. Registration of an image sequence is posed as a set of nonlinear optimization problems, each of which estimating $\mathbf{u}_i$ using the registration $\mathbf{u}_{i-1}$ of the previous frame as an initial solution. The discrepancy function $\mathcal{C}$ is usually chosen as the two-norm of the difference $\mathcal{D}$ between the texture image and the current one, warped towards the texture image, *i.e.* $\mathcal{D}(\mathbf{q}; \mathbf{u}_i) = \mathcal{I}_0(\mathbf{q}) - \mathcal{I}_i(\mathcal{W}(\mathbf{q}; \mathbf{u}_i))$, giving:

$$\mathcal{C}(\mathbf{u}_i) = \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{D}(\mathbf{q}; \mathbf{u}_i)\|^2 . \tag{1}$$

Other choices are possible for the cost function, such as the Mutual Information, see *e.g.* (Pluim et al, 2003; Meyer et al, 1997), or a criterion based on the gradient of images, see (Haber and Modersitzki, 2006).

Several algorithms have been proposed to minimize $\mathcal{C}$. We classify them in two groups: the Forward Additive algorithms and the Inverse Compositional ones.

2.1 Forward Additive Algorithms

Forward Additive Gauss-Newton (FA-GN). Using an additive update of the parameter vector, *i.e.* $\mathbf{u}_i \leftarrow \mathbf{u}_i + \boldsymbol{\delta}$, Gauss-Newton can be used in a straightforward manner for minimizing (1) or in conjunction with complexity tuning schemes as in (Bartoli and Zisserman, 2004; Lim and Yang, 2005) for the TPS warp. The local Gauss-Newton approximation to $\mathcal{C}$ is given by the first order Taylor expansion in $\boldsymbol{\delta}$ of each squared term in (1):

$$\mathcal{C}(\mathbf{u}_i + \boldsymbol{\delta}) \approx \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{D}(\mathbf{q}; \mathbf{u}_i) + \mathbf{g}(\mathbf{q}; \mathbf{u}_i)^{\mathsf{T}}\boldsymbol{\delta}\|^2. \tag{2}$$

The gradient vector $\mathbf{g}$ is the product of the image gradient vector and of the Jacobian matrix K of the warp, *i.e.* $\mathbf{g} = \nabla\mathcal{I}_i^{\mathsf{T}}\mathsf{K}$. The Gauss-Newton approximation induces a Linear Least Squares minimization problem in $\boldsymbol{\delta}$. Defining J as the Jacobian matrix of the error, obtained by stacking the gradient vectors $\mathbf{g}(\mathbf{q}; \mathbf{u}_i)^{\mathsf{T}}$ for all the pixels $\mathbf{q}$ in $\mathfrak{R}$, and $\mathbf{d} = \boldsymbol{\xi}_{\mathfrak{R}}(\mathcal{D}(\bullet; \mathbf{u}_i))$ the residual error vector, the solution is obtained through the normal equations:

$$\mathsf{H}\boldsymbol{\delta} = -\mathbf{b} \quad \text{with} \quad \mathsf{H} = \mathsf{J}^{\mathsf{T}}\mathsf{J} \quad \text{and} \quad \mathbf{b} = \mathsf{J}^{\mathsf{T}}\mathbf{d}. \tag{3}$$

The matrix H is the Gauss-Newton approximation to the Hessian matrix. Note that J, H and $\mathbf{d}$ depend on $\mathbf{u}_i$. The Jacobian matrix J must be recomputed at each iteration, implying that H must be recomputed and inverted as well.

Forward Additive ESM (FA-ESM). A second order approximation of $\mathcal{C}$ called ESM (Efficient Second-order Minimization), theoretically better than the Gauss-Newton one, is proposed in (Benhimane and Malis, 2004). Combined with an additive update of the parameters, it gives:

$$\mathcal{C}(\mathbf{u}_i + \boldsymbol{\delta}) \approx \sum_{\mathbf{q}\in\mathfrak{R}} \left\| \mathcal{D}(\mathbf{q};\mathbf{u}_i) + \tfrac{1}{2}\left(\mathbf{g}(\mathbf{q};\mathbf{u}_i) + \mathbf{g}(\mathbf{q};\mathbf{u}_0)\right)^{\mathsf{T}} \boldsymbol{\delta} \right\|^2 . \tag{4}$$

The ESM approximation has been shown to improve the convergence rate, compared to Gauss-Newton, without increasing the computation time per iteration since the gradient vectors $\mathbf{g}(\mathbf{q};\mathbf{u}_0)$ are constant.

2.2 Inverse Compositional Algorithms

The major drawback of the two above-presented methods is that the image gradient vector for each pixel in $\mathfrak{R}$ must be recomputed at each iteration. This is the most expensive step of the process. A major improvement was proposed by (Baker and Matthews, 2004) with the Inverse Compositional algorithm. The first key idea consists in switching the roles of the texture and of the current images:

$$\min_{\tilde{\mathbf{u}}} \sum_{\mathbf{q}\in\mathfrak{R}} \left\| \mathcal{I}_0\left(\mathcal{W}(\mathbf{q};\tilde{\mathbf{u}})\right) - \mathcal{I}_i\left(\mathcal{W}(\mathbf{q};\mathbf{u}_i)\right) \right\| . \tag{5}$$

The second idea is to update the current warp by composition:

$$\mathcal{W}(\,\bullet\,;\mathbf{u}_i) \leftarrow \mathcal{W}(\,\bullet\,;\mathbf{u}_i) \circ \mathcal{W}^{-1}(\,\bullet\,;\tilde{\mathbf{u}}). \tag{6}$$

Using Gauss-Newton for local registration leads to a constant Jacobian matrix and a constant Hessian matrix whose inverse is thus pre-computed. Of course, the inverted warp $\mathcal{W}^{-1}$ exists only if the considered set of warps is a group. For instance, homographic warps are used in the original Inverse Compositional algorithm (Baker and Matthews, 2004). In this case, inversion is obtained by inverting the associated 3×3 matrix and composition by multiplying those matrices. Several attempts have been made to relax the groupwise requirement for flexible models.

As proposed in (Gay-Bellile et al, 2006; Matthews and Baker, 2004; Romdhani and Vetter, 2003), these attempts usually consist in finding the best approximating warp for the pixels of interest in $\mathfrak{R}$:

$$\mathbf{u}_i \leftarrow \arg\min_{\mathbf{u}'_i} \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{W}(\mathcal{W}(\mathbf{q};\mathbf{u});\mathbf{u}_i) - \mathcal{W}(\mathbf{q};\mathbf{u}'_i)\|^2. \tag{7}$$

In (Matthews and Baker, 2004; Romdhani and Vetter, 2003), the warp is induced by a triangular mesh whose deformations are guided by a parameter vector. This minimization problem is usually solved in two steps. First the vertices in the current image are computed using the assumption of local rigidity. They usually are not in accordance with a model instance in *e.g.* the case of 3D Morphable Model (Gay-Bellile et al, 2006; Romdhani and Vetter, 2003). Second, the parameter update is recovered by minimizing a prediction error, *i.e.* the distance between the updated vertices and those induced by the parameters. This last step may be time consuming since nonlinear optimization is required. Warp inversion is approximated with first order Taylor expansion in (Matthews and Baker, 2004), while (Romdhani and Vetter, 2003) draws on triangular meshes to avoid linearization. By comparison, our method reverts and threads warps in closed-form: it does not require optimization.

Other methods have been proposed to obviate the shortcomings induced by non-groupwise warps. For instance, one may force the solution space to contain diffeomorphic warps (Joshi and Miller, 2000; Johnson and Christensen, 2002; Charpiat et al, 2005). The solution space thus constitutes a group. Requiring the warps to be diffeomorphisms may be an overly strong requirement. This is especially true when the deformations are not too important. Indeed, as noted in (Johnson and Christensen, 2002), with such deformations the estimated warps may be diffeomorphisms even though the solution space contains non-diffeomorphic warps. Besides, such approaches make the use of standard deformation models such as the TPS and the FFD generally impossible. This is one interesting point of the proposed approach in this paper: it proposes an efficient method which is built on top of the most common deformation models for image registration. Finally, enforcing the estimated warps to be diffeomorphisms is often achieved by adding supplementary constraints to an initial forward estimation algorithm. These constraints are generally impractical to design a fast inverse-compositional estimation algorithm.

3 Feature-Driven Registration

In this section, we present our principal contribution: the Feature-Driven framework. This framework, in which one directly acts on warp driving features, has two main advantages. First, it often is better balanced to tune feature positions, expressed in pixels, than coefficient vectors that may be difficult to interpret, as for the TPS or the FFD warps. Second, it allows one to use the efficient compositional framework in a straightforward manner. Indeed, warp composition and inversion cannot be directly done for non-groupwise warps. We propose empirical means for approximating warp composition and inversion through their driving features, called threading and reversion respectively. Our Feature-Driven framework is generic in the sense that it can be applied to almost any parametric warps such as the TPS or the FFD warps, as shown in §5.

3.1 Feature-Driven Warp Parameterization

Ignoring the set of parameters, a warp is an $\mathbb{R}^2 \to \mathbb{R}^2$ function usually parameterized by a set of n control points $\mathbf{p}_i = (p_i^x \;\; p_i^y)^{\mathsf{T}} \in \mathbb{R}^2$ for $i = 1,\ldots,n$. These control points are grouped in a vector $\mathbf{p} = \boldsymbol{\nu}(\mathsf{P}) \in \mathbb{R}^{2n}$ where $\mathsf{P} \in \mathbb{R}^{n\times 2}$ is the matrix defined by $\mathsf{P}^{\mathsf{T}} = (\mathbf{p}_1 \;\ldots\; \mathbf{p}_n)$. We write ω the warp in its natural parameterization. The warp ω is said to be linear when it can be defined as a linear combination of its control points:

$$\omega(\mathbf{q};\mathbf{p}) = \boldsymbol{\ell}_{\mathbf{q}}^{\mathsf{T}}\mathsf{P}, \tag{8}$$

where $\boldsymbol{\ell}_{\mathbf{q}} \in \mathbb{R}^n$ is a vector that depends on the point $\mathbf{q}$ and on the type of warp being considered. Note that the dependency on $\mathbf{q}$ of $\boldsymbol{\ell}_{\mathbf{q}}$ is usually non linear even if the warp is linear. The

control points are usually not interpolated. They just act as 'attractors' to the warp. It thus makes their interpretation difficult. The Feature-Driven concept is in fact a change of parameterization. The control points are replaced by a set of features that are interpolated by the warp. We call them the *driving features* and denote them $\mathbf{u}_0$ in the texture image and $\mathbf{v}$ in the current one (see figure 1). We denote $\mathcal{W}$ the Feature-Driven parameterization of the warp ω. Loosely speaking, matching the driving features between two images is equivalent to defining a warp since the warp can be used to transfer the driving features from one image to the other, while conversely, the warp can be computed from the driving features. Indeed, if the warp is linear with respect to its control points then it is always possible to find a matrix E such that:

$$\mathcal{W}(\mathbf{q};\mathbf{v}) = \boldsymbol{\ell}_{\mathbf{q}}^{\mathsf{T}}\mathsf{E}\mathsf{V}, \tag{9}$$

with $\mathsf{V} = \zeta(\mathbf{v})$, *i.e.* V equals to $\mathbf{v}$ reshaped on two columns. Matrix E can be pre-computed. Details on how the matrix E is obtained for the TPS and the FFD warps are given in sections §5.1 and §5.2 respectively. If we write $\boldsymbol{\mu}_{\mathbf{q}} = \mathsf{E}^{\mathsf{T}}\boldsymbol{\ell}_{\mathbf{q}}$ then the Feature-Driven parameterization of the warp $\mathcal{W}$ is given by:

$$\mathcal{W}(\mathbf{q};\mathbf{v}) = \boldsymbol{\mu}_{\mathbf{q}}^{\mathsf{T}}\mathsf{V}. \tag{10}$$

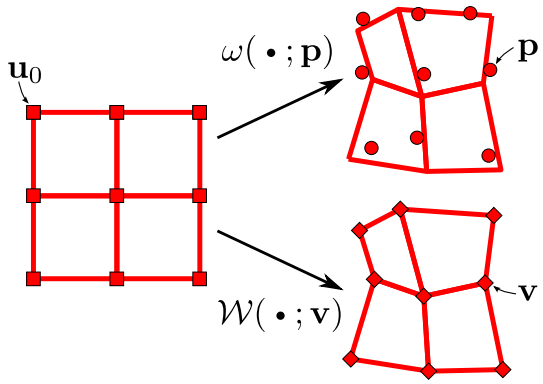


Figure 1: Illustration of the Feature-Driven parameterization. The vector $\mathbf{u}_0$ contains the features in the texture image (the centers). $\omega(\bullet;\mathbf{p})$ and $\mathcal{W}(\bullet;\mathbf{v})$ are two different representations of the same warp. The first one is parameterized with the 'natural' control points $\mathbf{p}$ while the second one is parameterized with the driving features $\mathbf{v}$.

Identity Warp. If we denote $\mathbf{u}_0$ the features in the texture image then $\mathcal{W}(\bullet;\mathbf{u}_0)$ is the identity warp[2], *i.e.* the warp that leaves the location of the features $\mathbf{u}_0$ unchanged.

3.2 Threading Warps

Given two sets of driving features, $\mathbf{v} = \boldsymbol{\nu}\left((\mathbf{v}_1 \ \dots \ \mathbf{v}_l)^{\mathsf{T}}\right)$ and $\mathbf{v}' = \boldsymbol{\nu}\left((\mathbf{v}'_1 \ \dots \ \mathbf{v}'_l)^{\mathsf{T}}\right)$, we want to find a third set $\mathbf{v}'' = \boldsymbol{\nu}\left((\mathbf{v}''_1 \ \dots \ \mathbf{v}''_l)^{\mathsf{T}}\right)$ defined such that threading the warps induced by $\mathbf{v}$ and $\mathbf{v}'$ results in the warp induced by $\mathbf{v}''$, as shown in figure 2. We propose a simple and computationally cheap way to do it, as opposed to previous work. This is possible thanks to the Feature-Driven parameterization. Our idea for threading warps is very simple: we apply the $\mathbf{v}'$ induced warp to the features $\mathbf{v}$; the resulting set of features is $\mathbf{v}''$. We thus define the warp threading operator, denoted $\square$, as:

$$\mathcal{W}(\bullet;\mathbf{v}) \square \mathcal{W}(\bullet;\mathbf{v}') \stackrel{\text{def}}{=} \mathcal{W}(\bullet;\mathbf{v}'') \qquad \text{with } \mathbf{v}'' = \mathcal{W}(\mathbf{v};\mathbf{v}'), \tag{11}$$

where $\mathcal{W}(\mathbf{v};\mathbf{v}')$ is meant to be applied to each feature in $\mathbf{v}$.

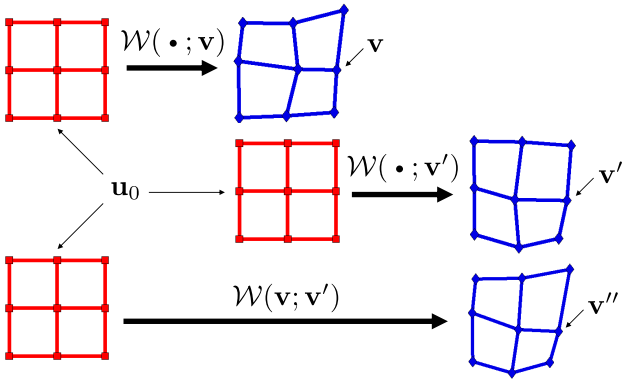


Figure 2: The Feature-Driven warp threading process: $\mathbf{v}''$ is defined by $\mathbf{v}'' = \mathcal{W}(\mathbf{v};\mathbf{v}')$.

Examples of our warp threading process are shown in figure 3. We synthesized two sets of driving features $\mathbf{v}$ and $\mathbf{v}'$ by randomly disturbing a 3×3 regular grid from its rest position $\mathbf{u}_0$. As expected, threading a warp $\mathcal{W}(\bullet;\mathbf{v})$ with the identity $\mathcal{W}(\bullet;\mathbf{u}_0)$ returns the original warp *i.e.* $\mathcal{W}(\bullet;\mathbf{v}) = \mathcal{W}(\bullet;\mathbf{v}) \square \mathcal{W}(\bullet;\mathbf{u}_0)$ and $\mathcal{W}(\bullet;\mathbf{v}) = \mathcal{W}(\bullet;\mathbf{u}_0) \square \mathcal{W}(\bullet;\mathbf{v})$.

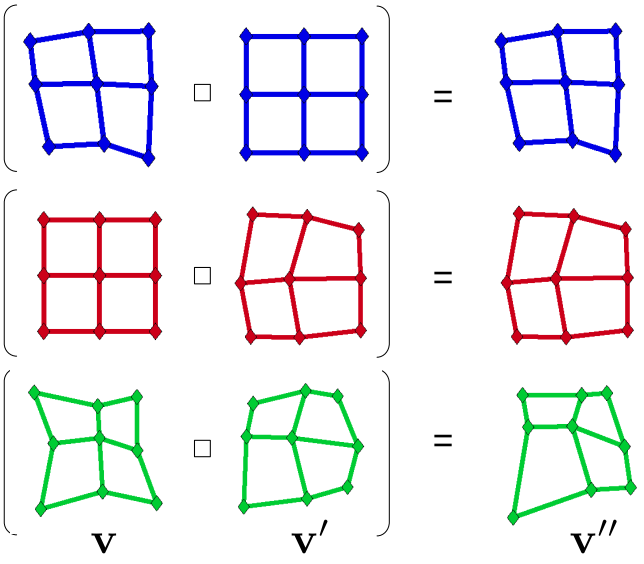


Figure 3: Examples for the warp threading process.

3.3 Reverting Warps

Given a set $\mathbf{v}$ of driving features, we want to determine the features $\mathbf{v}'$ such that the warp they induce is the reversion of the one induced by $\mathbf{v}$. This is illustrated in figure 4. As for the threading, our Feature-Driven framework yields a very simple solution. The idea is that applying the $\mathbf{v}'$ induced warp to $\mathbf{v}$

[2]Actually, it can also be a very close approximation, depending on how the matrix E is defined.

should give $\mathbf{u}_0$ *i.e.*, the fixed driving features in the texture image. We thus introduce the reversion operator $\diamond$ as:

$$\mathcal{W}(\,\bullet\,;\mathbf{v})^{\diamond} \stackrel{\text{def}}{=} \mathcal{W}(\,\bullet\,;\mathbf{v}') \qquad \text{with } \mathcal{W}(\mathbf{v};\mathbf{v}') = \mathbf{u}_0, \tag{12}$$

This amounts to solving an exactly determined linear system, the size of which is the number of driving features. Using equation (10), we obtain:

$$\mathsf{M}\mathsf{V}' = \mathsf{U}_0, \tag{13}$$

where $\mathsf{M} \in \mathbb{R}^{l\times l}$ is the matrix defined by $\mathsf{M}^{\mathsf{T}} = \begin{pmatrix}\boldsymbol{\mu}_{\mathbf{v}_1} & \cdots & \boldsymbol{\mu}_{\mathbf{v}_l}\end{pmatrix}$, $\mathsf{V}' = \zeta(\mathbf{v}')$ and $\mathsf{U}_0 = \zeta(\mathbf{u}_0)$. The driving features $\mathbf{v}'$ of the reverted warp are thus given by:

$$\mathbf{v}' = \boldsymbol{\nu}\left(\mathsf{M}^{-1}\mathsf{U}_0\right). \tag{14}$$

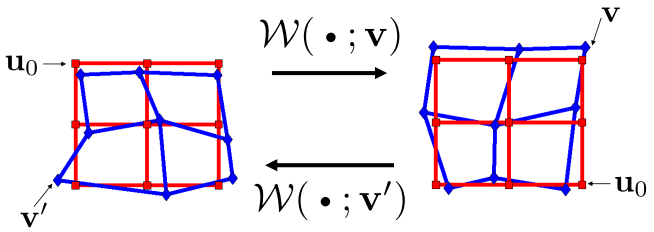


Figure 4: The Feature-Driven warp reversion process: $\mathbf{v}'$ is defined such that $\mathcal{W}(\mathbf{v};\mathbf{v}') = \mathbf{u}_0$.

Examples of the reverting process are shown in figure 5. We synthesized driving features $\mathbf{v}$ by randomly disturbing a 3×3 regular grid from its rest position $\mathbf{u}_0$. The driving features $\mathbf{v}'$ result from reverting the warp $\mathcal{W}(\bullet;\mathbf{v})$. Threading warps $\mathcal{W}(\bullet;\mathbf{v})$ and $\mathcal{W}(\bullet;\mathbf{v}')$ introduce a new set of driving features $\mathbf{v}''$. As expected, the features $\mathbf{v}''$ are similar to the original grid $\mathbf{u}_0$ with an average residual error of 10^{-13} pixels.

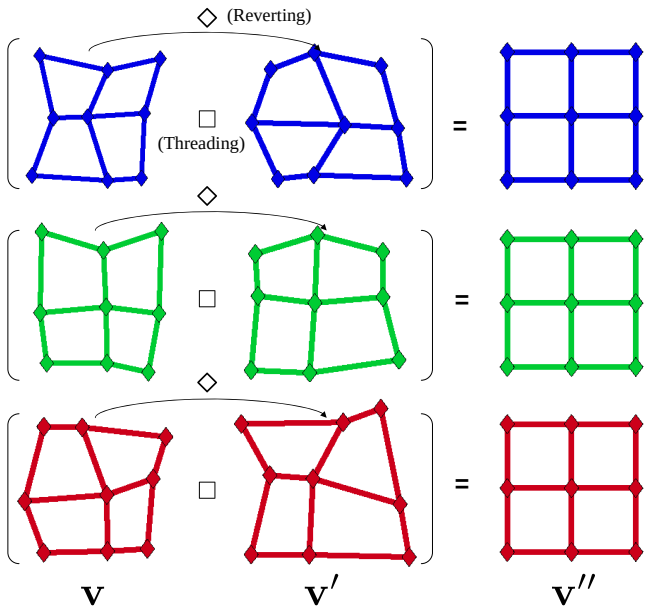


Figure 5: Illustration of the warp reversion process on three examples.

3.4 Compositional Feature-Driven Registration

Relying on the Feature-Driven parameterization properties, we extend compositional algorithms to non-groupwise warps. The following three steps are repeated until convergence, as shown in figure 6:

- **Step 1: Warping.** The current driving features $\mathbf{u}_i$ are used to warp the input image $\mathcal{I}_i$, thereby globally registering it to the texture image by creating the warped image $\mathcal{I}_W$:

$$\mathcal{I}_W(\mathbf{q}) \stackrel{\text{def}}{=} \mathcal{I}_i(\mathcal{W}(\mathbf{q};\mathbf{u}_i)). \tag{15}$$

- **Step 2: Local registration.** The driving features $\mathbf{u}$ are estimated in the warped image $\mathcal{I}_W$. Several algorithms can be used. They are described in §4.1 and §4.2. Note that for the Inverse Compositional algorithm, warp reversion is done at this step, based on equation (12).

- **Step 3: Updating.** The current driving features $\mathbf{u}_i$ and those in the warped image $\mathbf{u}$ are combined by threading the warps using equation (11), to update the driving features $\mathbf{u}_i$ in the current image.

Note that in previous work (Gay-Bellile et al, 2006; Matthews and Baker, 2004; Romdhani and Vetter, 2003) a preliminary step is required before applying the update rule, as reviewed in §2.2. In comparison, our Feature-Driven framework makes it naturally included into the third step.

Illumination changes are handled by globally normalizing the pixel values in the texture and the warped images at each iteration. Another approach could be used such as the light-invariant approach of (Pizarro and Bartoli, 2007).

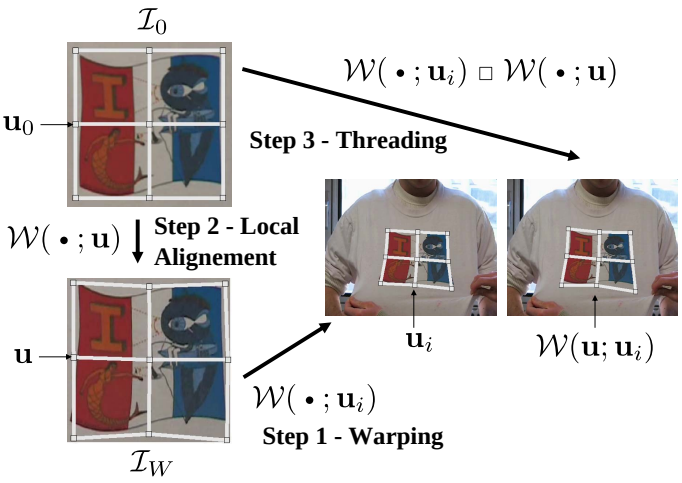


Figure 6: The three steps of the Compositional Feature-Driven registration.

4 Local Registration Algorithms

In the literature, there are two ways for estimating local registration: the Forward and the Inverse approaches (Baker and Matthews, 2004). The former evaluates directly the warp which aligns the texture image $\mathcal{I}_0$ with the warped image $\mathcal{I}_W$. The latter computes the warp which aligns the warped with the texture image and then inverts the warp. They are both compatible

with approximations of the cost function such as Gauss-Newton, ESM, learning-based, *etc.* We describe in details the Inverse Gauss-Newton and the Forward Learning-based local registration steps.

4.1 Local Registration with Gauss-Newton

Combining an Inverse local registration with a Gauss-Newton approximation of the cost function is efficient since this combination makes invariant the approximated Hessian matrix used in the normal equations to be solved at each iteration. We cast this approach in the Feature-Driven framework, making it possible to extend Inverse Compositional registration to the TPS and the FFD warps.

In the Inverse Compositional framework, local registration is achieved by minimizing the local discrepancy error:

$$\mathcal{C}_l(\tilde{\mathbf{u}}) = \sum_{\mathbf{q}\in\mathfrak{R}} \|\mathcal{I}_0(\mathcal{W}(\mathbf{q};\tilde{\mathbf{u}})) - \mathcal{I}_W(\mathbf{q})\|^2. \tag{16}$$

Using Gauss-Newton as local registration engine, the gradient vector is the product of the texture image gradient vector and of the constant Jacobian matrix K of the warp: $\mathbf{g} = \nabla\mathcal{I}_0^\mathsf{T}\mathsf{K}$. Matrix K is given in §5. The Jacobian matrix of this least squares cost is thus constant. The Hessian matrix H and its inverse are computed off-line. However, the driving features $\tilde{\mathbf{u}}$ are located on the reference image $\mathcal{I}_0$. They must be located on the warped image $\mathcal{I}_W$ for being used in the update. We use our warp reversion process for finding the driving features $\mathbf{u}$ on the warped image *i.e.*, $\mathbf{u}$ such that $\mathcal{W}(\tilde{\mathbf{u}};\mathbf{u}) = \mathbf{u}_0$. An overview of Feature-Driven Inverse Gauss-Newton registration is shown in table 1.

Off-line

- Evaluate the gradient $\nabla\mathcal{I}_0$ of the reference image
- Evaluate the constant Jacobian K of the warp
- Compute the Jacobian matrix J of the cost function
- Compute the pseudo Hessian $\mathsf{H} = \mathsf{J}^\mathsf{T}\mathsf{J}$ and its inverse H^{-1}

On-line

- Compute the error vector $\mathbf{d} = \boldsymbol{\xi}_\mathfrak{R}(\mathcal{I}_0 - \mathcal{I}_W)$
- Compute $\mathbf{b} = \mathsf{J}^\mathsf{T}\mathbf{d}$
- Estimate $\tilde{\mathbf{u}} = \mathbf{u}_0 - \mathsf{H}^{-1}\mathbf{b}$
- Find the driving features $\mathbf{u}$ such that $\mathcal{W}(\tilde{\mathbf{u}};\mathbf{u}) = \mathbf{u}_0$
- Update by threading: $\mathcal{W}(\bullet;\mathbf{u}_i) \leftarrow \mathcal{W}(\bullet;\mathbf{u}_i) \,\square\, \mathcal{W}(\bullet;\mathbf{u})$

Table 1: Overview of our Feature-Driven Inverse Compositional Gauss-Newton registration.

4.2 Learning-Based Local Registration

Learning-based methods model the relationship between the local increment $\boldsymbol{\delta}$ and the intensity discrepancy $\mathbf{d}$ with an interaction function f:

$$\boldsymbol{\delta} = f(\mathbf{d}). \tag{17}$$

The interaction function is often approximated using a linear model, *i.e.* $f(\mathbf{d}) = \mathsf{F}\mathbf{d}$ where F is called the interaction matrix. This relationship is valid locally around the texture image parameters $\mathbf{u}_0$. Compositional algorithms are thus required, as in (Jurie and Dhome, 2002) for homographic warps. The Feature-Driven framework naturally extends this approach to non-groupwise warps. However in (Cootes et al, 1998) the assumption is made that the domain where the linear relationship is valid covers the whole set of registrations. They thus apply their interaction function around the current parameters, avoiding the warping and the composition steps. This does not appear to be a valid choice in practice.

The interaction function is learned from artificially perturbed texture images $\mathcal{A}_j$. They are obtained through random perturbations of the reference parameter $\mathbf{u}_0$. In the literature, linear and non linear interaction functions are used. They are learned with different regression algorithms such as Least Squares (LS) (Cootes et al, 1998; Jurie and Dhome, 2002), Support Vector Machines (SVM) or Relevance Vector Machines (RVM) (Agarwal and Triggs, 2006). Details are given below for a linear interaction function, *i.e.* an interaction matrix, learned through Least Squares regression. Table 2 summarizes the steps of learning-based local registration.

Off-line

- Learn the interaction function f

On-line

- Compute the error vector $\mathbf{d} = \boldsymbol{\xi}_\mathfrak{R}(\mathcal{I}_0 - \mathcal{I}_W)$
- Compute $\mathbf{u} = \mathbf{u}_0 + f(\mathbf{d})$
- Update by threading: $\mathcal{W}(\bullet;\mathbf{u}_i) \leftarrow \mathcal{W}(\bullet;\mathbf{u}_i) \,\square\, \mathcal{W}(\bullet;\mathbf{u})$

Table 2: Overview of our Learning-based registration.

Generating training data with a Feature-Driven Warp. The driving features in the texture image are disturbed from their rest position $\mathbf{u}_0$ with randomly chosen directions $\boldsymbol{\theta}_j$ and magnitudes $\mathbf{r}_j$:

$$\mathbf{u}_j = \mathbf{u}_0 + \boldsymbol{\delta}_j \qquad \text{with } \boldsymbol{\delta}_j = \begin{pmatrix} \mathbf{r}_j \odot \cos(\boldsymbol{\theta}_j) \\ \mathbf{r}_j \odot \sin(\boldsymbol{\theta}_j) \end{pmatrix}, \tag{18}$$

where $\cos(\boldsymbol{\theta}_j)$ and $\sin(\boldsymbol{\theta}_j)$ are meant to be applied to all the elements of $\boldsymbol{\theta}_j$ and $\odot$ denotes the element-wise product. The magnitude is clamped between a lower and an upper bound, determining the area of validity of the interaction matrix to be learned. For a Feature-Driven warp, fixing this magnitude is straightforward since the driving features are expressed in pixels. It can be much more complex when the parameters are difficult to interpret such as the usual coefficients of the TPS and the FFD warps. There are two ways to synthesize images:

$$\mathcal{A}_j(\mathbf{q}) \leftarrow \mathcal{I}_0(\mathcal{W}(\mathbf{q};\mathbf{u}_j)^\diamond) \tag{19}$$

or

$$\mathcal{A}_j(\mathbf{q}) \leftarrow \mathcal{I}_0(\arg\min_\mathbf{q} \|\mathcal{W}(\mathbf{q};\mathbf{u}_j) - \mathbf{q}\|). \tag{20}$$

The former requires warp inversion whereas the latter requires a cost optimization, per-pixel. In our experiments, we use equation (19). Our Feature-Driven warp reversion process is thus used to warp the texture image. Training data generation with a Feature-Driven warp is illustrated in figure 7.

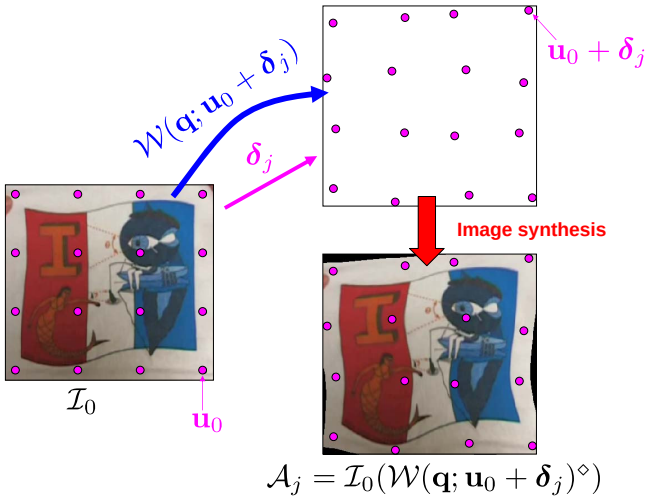


Figure 7: Generating training data with a Feature-Driven warp.

Learning. The residual vector is computed for the pixels of interest in $\mathfrak{R}$:

$$\mathbf{d}_j = \boldsymbol{\xi}_{\mathfrak{R}}\left(\mathcal{I}_0 - \mathcal{A}_j\right). \tag{21}$$

The training data are gathered in matrices $\mathsf{D} = (\boldsymbol{\delta}_1 \ \dots \ \boldsymbol{\delta}_m) \in \mathbb{R}^{|\mathfrak{R}|\times m}$ and $\mathsf{L} = (\mathbf{d}_1 \ \dots \ \mathbf{d}_m) \in \mathbb{R}^{|\mathfrak{R}|\times m}$. The interaction matrix $\mathsf{F} \in \mathbb{R}^{|\mathfrak{R}|\times|\mathfrak{R}|}$ is computed by minimizing a Linear Least Squares error in the image space, expressed in pixel value unit, giving:

$$\mathsf{F} = \left(\mathsf{L}\mathsf{D}^\mathsf{T}(\mathsf{D}\mathsf{D}^\mathsf{T})^{-1}\right)^\dagger. \tag{22}$$

This is one of the two possibilities for learning the interaction matrix. The other possibility is dual. It minimizes an error in the parameter space, *i.e.* expressed in pixels. The two approaches have been experimentally compared. Learning the interaction matrix in the image space give the best results. Thereafter, we use this option.

A piecewise linear interaction function. Experiments show that a linear approximation of the relationship between the local increment $\boldsymbol{\delta}$ and the intensity discrepancy $\mathbf{d}$, though computationally efficient, does not always give satisfying results. The drawback is that if the interaction matrix covers a large domain of deformation magnitudes, the registration accuracy is spoiled. On the other hand, if the matrix is learned for small deformations only, the convergence basin is dramatically reduced. Using a nonlinear interaction function learned through RVM or SVM partially solves this issue. We use a simple piecewise linear relationship as interaction function. It means that we learn not only one but a series $\mathsf{F}_1, \dots, \mathsf{F}_\kappa$ of interaction matrices, each of them covering a different range of displacement magnitudes. The interaction function is thus of the form $f(\mathbf{d}) = \sum_{i=1}^{\kappa} a_i \mathsf{F}_i$. More details are given in appendix A.

5 Feature-Driven Warps

In this section, we specialize the generic Feature-Driven parameterization presented in §3.1 for two types of warps: the TPS and the FFD warps. Since the representational power of the TPS warp and of the FFD warp are equivalent (see experiments in §6.1), we focused our experiments on the TPS warp. However, it is important to show how the FFD warp can actually be used in the Feature-Driven framework. In particular, we show how the standard FFD model can be extended in order to be compatible with the warp reversion operation.

5.1 The Feature-Driven Thin-Plate Spline Warp

5.1.1 Definition

Ignoring the parameters, a TPS $\bar{\omega}_\mathsf{T}$ is an $\mathbb{R}^2 \to \mathbb{R}$ function. It is the Radial Basis Function that minimizes the integral bending energy. In its natural parameterization, a TPS is driven by a set of $l+3$ weights $\bar{p}_k \in \mathbb{R}$. These weights are grouped in a vector of parameters $\bar{\mathbf{p}} \in \mathbb{R}^{l+3}$. The evaluation of a TPS at the point $\mathbf{q}^\mathsf{T} = (x \ y)$ is given by:

$$\bar{\omega}_\mathsf{T}(\mathbf{q};\bar{\mathbf{p}}) = \sum_{i=1}^{l} \bar{p}_i \rho\left(d^2(\mathbf{q},\mathbf{c}_i)\right) + \bar{p}_{l+1}x + \bar{p}_{l+2}y + \bar{p}_{l+3}. \tag{23}$$

The l 2D points $\mathbf{c}_k$ are called the centers. They are also the driving features in the texture image. They can be located at any place but, in practice, we place them on a regular grid. The function d^2 gives the squared euclidean distance between its two arguments. The function ρ is the TPS basis function and is defined by $\rho(r) = r^2\log(r)$ for $r > 0$ and $\rho(0) = 0$. In matrix form, equation (23) is equivalent to:

$$\bar{\omega}_\mathsf{T}(\mathbf{q};\bar{\mathbf{p}}) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T}\bar{\mathbf{p}}, \tag{24}$$

with

$$\boldsymbol{\ell}_\mathbf{q}^\mathsf{T} = \begin{pmatrix}\rho(d^2(\mathbf{q},\mathbf{c}_1)) & \cdots & \rho(d^2(\mathbf{q},\mathbf{c}_l)) & \mathbf{q}^\mathsf{T} & 1\end{pmatrix} \in \mathbb{R}^{l+3}.$$

Standard $\mathbb{R}^2 \to \mathbb{R}^2$ TPS warps are obtained by replacing the scalar weights $\bar{p}_i$ by the control points $\mathbf{p}_k = (p_k^x \ p_k^y) \in \mathbb{R}^2$. The control points are grouped in a single matrix of parameters $\mathsf{P} \in \mathbb{R}^{(l+3)\times 2}$ defined by $\mathsf{P}^\mathsf{T} = (\mathbf{p}_1 \ \dots \ \mathbf{p}_{l+3})$. The TPS warp is thus defined by:

$$\omega_\mathsf{T}(\mathbf{q};\mathbf{p}) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T}\mathsf{P}, \qquad \text{with } \mathbf{p} = \boldsymbol{\nu}(\mathsf{P}). \tag{25}$$

5.1.2 Feature-Driven Parameterization

The Feature-Driven parameterization of the TPS warp consists in replacing the control points by some features (*i.e.* points) in the current image. A point $\mathbf{a}_k^\mathsf{T} = (a_k^x \ a_k^y) \in \mathbb{R}^2$ is assigned to each center $\mathbf{c}_k$ defined in the texture image. The features $\mathbf{a}_k$ are grouped in a single matrix $\mathsf{A} \in \mathbb{R}^{l\times 2}$. Similarly, the centers $\mathbf{c}_k$ are grouped in a matrix $\mathsf{C} \in \mathbb{R}^{l\times 2}$. Following (Bookstein, 1989), the control points of a TPS can be determined from the correspondences $\mathbf{c}_k \leftrightarrow \mathbf{a}_k$:

$$\omega_\mathsf{T}(\mathbf{c}_k;\mathbf{p}) = \mathbf{a}_k \qquad \forall k \in \{1,\dots,l\}, \tag{26}$$

while enforcing the 3 'side-conditions' ensuring that the TPS has square integrable second derivatives (more details can be found in (Wahba, 1990)):

$$\sum_{i=1}^{l} x_i \mathbf{p}_i = \mathbf{0}_{2\times 1}, \quad \sum_{i=1}^{l} y_i \mathbf{p}_i = \mathbf{0}_{2\times 1}, \quad \sum_{i=1}^{l} \mathbf{p}_i = \mathbf{1}_{2\times 1}. \tag{27}$$

Combining these $l+3$ conditions in a single matrix gives the following exactly determined linear system:

$$\mathsf{M}_\lambda \mathsf{P} = \begin{pmatrix} \mathsf{A} \\ \mathbf{0}_{3\times 1} \end{pmatrix}, \tag{28}$$

with $\mathsf{M}_\lambda \in \mathbb{R}^{(l+3)\times(l+3)}$ the matrix defined by:

$$\mathsf{M}_\lambda = \begin{pmatrix} \mathsf{N}_\lambda & \mathsf{Q} \\ \mathsf{Q}^\mathsf{T} & \mathbf{0}_{3\times 3} \end{pmatrix}, \tag{29}$$

with $\mathsf{N}_\lambda = \mathsf{N} + \lambda \mathrm{I}_l$, $\mathsf{N}^\mathsf{T} = \begin{pmatrix} \boldsymbol{\ell}_{\mathbf{c}_1}^\mathsf{T} & \cdots & \boldsymbol{\ell}_{\mathbf{c}_l}^\mathsf{T} \end{pmatrix}$ and $\mathsf{Q} = \begin{pmatrix} \mathsf{C} & \mathbf{1}_{l\times 1} \end{pmatrix} \in \mathbb{R}^{l\times 3}$. Adding $\lambda \mathrm{I}_l$ to N acts as a regularizer. Determining the control points P from the equation (28) can be done in a straightforward manner as the solution of an exactly determined linear system. The resulting matrix of control points, denoted P_λ, is a nonlinear function of the regularization parameter λ and a linear function of the features A:

$$\mathsf{P}_\lambda = \mathsf{M}_\lambda^{-1} \begin{pmatrix} \mathsf{A} \\ \mathbf{0}_{3\times 1} \end{pmatrix}. \tag{30}$$

P_λ is a linear 'back-projection' of the feature matrix A. It can be computed efficiently using the blockwise matrix inversion formulas:

$$\mathsf{P}_\lambda = \mathsf{E}_\lambda \mathsf{A} \tag{31}$$

with:

$$\mathsf{E}_\lambda = \begin{pmatrix} \mathsf{N}_\lambda^{-1} \left(\mathrm{I}_l - \mathsf{Q} \left(\mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \mathsf{Q} \right) \mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \right) \\ \left(\mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \mathsf{Q} \right)^{-1} \mathsf{Q}^\mathsf{T} \mathsf{N}_\lambda^{-1} \end{pmatrix}. \tag{32}$$

This expression has the advantages of separating λ and A and introduces units: while P_λ has no obvious unit, A in general has (*e.g.* pixels, meters). Finally, if we replace the natural parameters $\mathbf{p}$ in the definition of the TPS warp ω_T (equation (25)) by their expression given in the equation (32), we get the Feature-Driven parameterization of the TPS warp, denoted $\mathcal{W}_\mathsf{T}$:

$$\mathcal{W}_\mathsf{T}(\mathbf{q}; \mathbf{a}, \lambda) = \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E}_\lambda \mathsf{A}, \qquad \text{with } \mathbf{a} = \boldsymbol{\nu}(\mathsf{A}). \tag{33}$$

We use the notation $\mathcal{W}_\mathsf{T}(\mathbf{q}; \mathbf{a})$ for $\mathcal{W}_\mathsf{T}(\mathbf{q}; \mathbf{a}, 10^{-4})$. We choose $\lambda = 10^{-4}$ to ensure good numerical conditioning of the matrix N_λ.

Jacobian matrix of the warp. The Jacobian matrix of the warp is needed by the Gauss-Newton based algorithms for local registration (see *e.g.*, §2.1 or §4.1). We denote K_T the Jacobian matrix of the TPS warp evaluated at the point $\mathbf{q}$. It is defined by $\mathsf{K}_\mathsf{T} = \frac{\partial \mathcal{W}_\mathsf{T}}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \in \mathbb{R}^{2\times 2(l+3)}$ and is given by:

$$\begin{aligned} \mathsf{K}_\mathsf{T} &= \begin{pmatrix} \frac{\partial \mathcal{W}_\mathsf{T}^x}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \\ \frac{\partial \mathcal{W}_\mathsf{T}^y}{\partial \mathbf{a}}(\mathbf{q}; \mathbf{a}) \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathcal{W}_\mathsf{T}^x}{\partial \mathbf{a}^x}(\mathbf{q}; \mathbf{a}) & \mathbf{0}_{1\times(l+3)} \\ \mathbf{0}_{1\times(l+3)} & \frac{\partial \mathcal{W}_\mathsf{T}^y}{\partial \mathbf{a}^y}(\mathbf{q}; \mathbf{a}) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E}_\lambda & \mathbf{0}_{1\times(l+3)} \\ \mathbf{0}_{1\times(l+3)} & \boldsymbol{\ell}_\mathbf{q}^\mathsf{T} \mathsf{E}_\lambda \end{pmatrix}, \end{aligned} \tag{34}$$

where $\mathcal{W}_\mathsf{T}^x$ and $\mathcal{W}_\mathsf{T}^y$ are the first and the second coordinates of the warp $\mathcal{W}_\mathsf{T}$ and $\mathsf{A} = (\mathbf{a}^x \ \mathbf{a}^y)$.

5.2 The Feature-Driven Free-Form Deformation

Tensor-product B-Splines are a particular model of Free-Form Deformations. They are a general model of polynomial functions which have been proved to be useful for image registration (Rueckert et al, 1999). Even if there is a wide variety of B-Splines (with various degrees for the polynomial basis or by choosing exotic knot sequences), we limit our study to the case of the Uniform Cubic B-Splines since it best matches the needs of image registration. For the sake of simplicity, we will abbreviate it FFD.

5.2.1 Definition

Monodimensional case. Ignoring the parameters, a monodimensional FFD $\bar{\omega}_\mathsf{F}$ is an $\mathbb{R} \to \mathbb{R}$ function defined as a linear combination of the basis functions N_i weighted by the scalars $\bar{p}_k$ called the weights:

$$\bar{\omega}_\mathsf{F}(x; \bar{\mathbf{p}}) = \sum_{i=1}^{m} \bar{p}_i N_i(x), \tag{35}$$

where $\bar{\mathbf{p}} \in \mathbb{R}^m$ is the vector that contains all the weights $\bar{p}_k$. The basis functions are defined using a knot sequence, *i.e.* a non-decreasing sequence $k_1 < \ldots < k_{m+4}$. The FFD is said to be uniform when the knot sequence is uniform, *i.e.* all the knot intervals $[k_i, k_{i+1}]$ have the same length s. In this case, the basis functions N_i are defined by using four polynomials of degree three, the blending functions (see figure 8(a) for an illustration):

$$N_i(x) = \begin{cases} b_1(x) = \frac{1}{6}\hat{x}^3 & \\ & \text{if } x \in [k_i, k_{i+1}] \\ b_2(x) = \frac{1}{6}\left(-3\hat{x}^3 + 3\hat{x}^2 + 3\hat{x} + 1\right) & \\ & \text{if } x \in [k_{i+1}, k_{i+2}] \\ b_3(x) = \frac{1}{6}\left(3\hat{x}^3 - 6\hat{x}^2 + 4\right) & \\ & \text{if } x \in [k_{i+2}, k_{i+3}] \\ b_4(x) = \frac{1}{6}\left(-\hat{x}^3 + 3\hat{x}^2 - 3\hat{x} + 1\right) & \\ & \text{if } x \in [k_{i+3}, k_{i+4}] \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

where $\hat{x}$ is the normalized abscissa of x defined as $\hat{x} = \frac{x - k_I}{s}$ for $x \in [k_I, k_{I+1}]$.

Domain. We can see from equation (35) that an FFD is non-zero only over the interval $[k_1, k_{m+4}]$. However, it is common practice to reduce the domain to $[k_4, k_{m+1}]$. By doing so, there are always exactly 4 non-zero basis functions on each knot interval, as figure 8(b) illustrates.

FFD warp. The standard $\mathbb{R}^2 \to \mathbb{R}^2$ FFD warp is obtained as the two-way tensor-product of monodimensional FFDs. Using its natural parameterization, the evaluation of the FFD warp ω_F at the point $\mathbf{q} = (x \ y)^\mathsf{T}$ is given by:

$$\omega_\mathsf{F}(\mathbf{q}; \mathbf{p}) = \sum_{j=1}^{n} \sum_{i=1}^{m} \mathbf{p}_k N_i(x) N_j(y), \quad \text{with } k = (j-1)m + i. \tag{37}$$

The mn control points $\mathbf{p}_k$ are grouped in the vector $\mathbf{p} \in \mathbb{R}^{2mn}$ that is defined as $\mathbf{p} = \boldsymbol{\nu}(\mathsf{P})$ where $\mathsf{P} \in \mathbb{R}^{mn\times 2}$ is the matrix given by $\mathsf{P}^\mathsf{T} = (\mathbf{p}_1 \ \ldots \ \mathbf{p}_{mn})$. The control points of an FFD

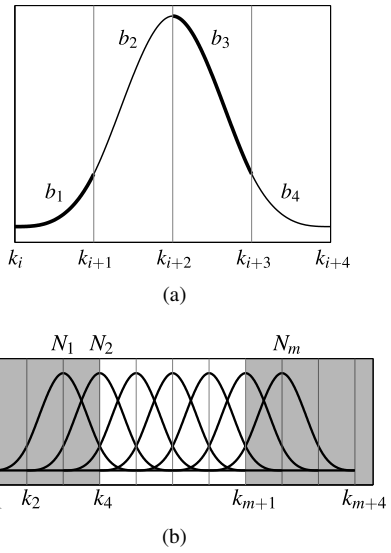


Figure 8: (a) The basis functions of an FFD are bell-shaped curves with bounded support. They are defined using 4 polynomial pieces of degree three: b_1, b_2, b_3 and b_4. (b) The usual (or natural) definition domain of an FFD is represented by the non-grayed part.

warp are not more meaningful than the ones of a TPS warp. They are not interpolated: they just act as 'attractors' to the warp. Equation (37) can be rewritten in matrix form:

$$\omega_{\mathsf{F}}(\mathbf{q};\mathbf{p}) = \boldsymbol{\ell}_{\mathbf{q}}^{\mathsf{T}}\mathsf{P}, \tag{38}$$

where $\boldsymbol{\ell}_{\mathbf{q}} \in \mathbb{R}^{mn}$ is the vector defined by:

$$\boldsymbol{\ell}_{\mathbf{q}}^{\mathsf{T}} = (N_1(x)N_1(y) \ \ldots \ N_m(x)N_1(y) \ \ldots \ N_m(x)N_n(y)). \tag{39}$$

5.2.2 Feature-Driven Parameterization

The Feature-Driven parameterization of the FFD warp is similar to the one of the TPS warp in the sense that it makes the warp driven by features expressed in pixels in both the texture and the current images. The centers of the TPS warp were used as features in the texture image. Such centers do not exist for FFD warps. We thus introduce a set of points $\mathbf{c}_k$ that will be used as features in the texture image. We call these points *centers* for consistency with the TPS warps. We use $l = mn$ centers located on a regular grid. A feature $\mathbf{a}_k$ in the current image is associated to every center $\mathbf{c}_k$. The control points $\mathbf{p}$ of the FFD warp can be determined from the correspondences $\mathbf{c}_k \leftrightarrow \mathbf{a}_k$ by enforcing the following constraints:

$$\omega_{\mathsf{F}}(\mathbf{c}_k;\mathbf{p}) = \mathbf{a}_k, \qquad \forall k \in \{1,\ldots,l\}. \tag{40}$$

Since the number of features is equal to the number of degrees of freedom of the FFD warp, the determination of the parameters from the features can be carried out with an exactly determined linear system:

$$\mathsf{MP} = \mathsf{A}, \tag{41}$$

with $\mathsf{M}^{\mathsf{T}} = \begin{pmatrix}\boldsymbol{\ell}_{\mathbf{c}_1} & \ldots & \boldsymbol{\ell}_{\mathbf{c}_l}\end{pmatrix} \in \mathbb{R}^{l\times l}$, $\mathsf{P} = \zeta(\mathbf{p}) \in \mathbb{R}^{l\times 2}$ and $\mathsf{A}^{\mathsf{T}} = (\mathbf{a}_1 \ \ldots \ \mathbf{a}_{mn}) \in \mathbb{R}^{2\times l}$. The solution of the linear system (41) can be written $\mathsf{P} = \mathsf{EA}$ where $\mathsf{E} = \mathsf{M}^{-1}$. The existence of the matrix E is guaranteed if the Schoenberg-Whitney conditions are satisfied (see (De Boor, 2001)) as it is the case when the centers are located on a regular grid. Note that the matrix E can be pre-computed. Finally, the Feature-Driven parameterization of the FFD warp, denoted $\mathcal{W}_{\mathsf{F}}$, is given by replacing the natural parameters $\mathbf{p}$ in equation (38) with their expression in function of the features $\mathbf{a} = \boldsymbol{\nu}(\mathsf{A}) \in \mathbb{R}^{2l}$:

$$\mathcal{W}_{\mathsf{F}}(\mathbf{q};\mathbf{a}) = \boldsymbol{\ell}_{\mathbf{q}}^{\mathsf{T}}\mathsf{EA}. \tag{42}$$

Jacobian matrix of the warp. The Jacobian matrix $\mathsf{K}_{\mathsf{F}} \in \mathbb{R}^{2\times 2l}$ for FFD warps can be computed following exactly the same reasoning as for the TPS warp:

$$\begin{aligned}\mathsf{K}_{\mathsf{F}} &= \begin{pmatrix}\frac{\partial \mathcal{W}_{\mathsf{F}}^x}{\partial \mathbf{a}}(\mathbf{q};\mathbf{a}) \\ \frac{\partial \mathcal{W}_{\mathsf{F}}^y}{\partial \mathbf{a}}(\mathbf{q};\mathbf{a})\end{pmatrix} = \begin{pmatrix}\frac{\partial \mathcal{W}_{\mathsf{F}}^x}{\partial \mathbf{a}^x}(\mathbf{q};\mathbf{a}) & \mathbf{0}_{1\times l} \\ \mathbf{0}_{1\times l} & \frac{\partial \mathcal{W}_{\mathsf{F}}^y}{\partial \mathbf{a}^y}(\mathbf{q};\mathbf{a})\end{pmatrix} \\ &= \begin{pmatrix}\boldsymbol{\ell}_{\mathbf{q}}^{\mathsf{T}}\mathsf{E} & \mathbf{0}_{1\times l} \\ \mathbf{0}_{1\times l} & \boldsymbol{\ell}_{\mathbf{q}}^{\mathsf{T}}\mathsf{E}\end{pmatrix}.\end{aligned} \tag{43}$$

5.2.3 Extrapolation

The computations involved in the warp reversion operation (see section §3.3) can lead to evaluate a warp outside of its natural definition domain. More precisely, in equation (11), nothing ensures that the features of the vector $\mathbf{v}$ lies in the domain of the warp. While this is not a problem with the TPS warp whose domain is infinite, extra work need to be done with the FFD warp. Indeed, with the previous definition, it is possible to evaluate an FFD warp outside of its natural domain but it is meaningless since it collapses to 0. In this section, we propose a new method to extrapolate an FFD warp outside of its domain making it virtually infinite.

The principle of the method is simple: a linear extension is added to the basis that crosses the boundaries of the domain (with some extra conditions of continuity and differentiability). While this seems almost trivial in the monodimensional case, it is less simple in two dimensions, *i.e.* for warps. Our strategy consists in defining the extension in 1D and, then, propagate it to the 2D case using the usual tensor-product.

We present the extrapolation approach in the monodimensional case and for the leftmost boundary of the domain (*i.e.*, the knot k_4). The four non-zero bases that cross this boundary are N_1, N_2, N_3 and N_4. Our idea is to drop the part of these bases that are outside the domain and to replace them with a linear extension. We call N_1^e, N_2^e, N_3^e and N_4^e the bases resulting from this process. In addition to be linear, we enforce the following constraints in order to preserve continuity and differentiability:

$$\left.\begin{aligned} N_i^e(k_4) &= N_i(k_4) \\ \frac{\partial N_i^e}{\partial x}(k_4) &= \frac{\partial N_i}{\partial x}(k_4)\end{aligned}\right\} \forall i \in 1,\ldots,4. \tag{44}$$

For the sake of simplicity and without loss of generality, we consider that the leftmost boundary coincides with zero ($k_4 = 0$) and that the length of the knot intervals is consistently one

($s = 1$). Under all these constraints, it follows that:

$$N_1^e(x) = \begin{cases} -\frac{x}{2} + \frac{1}{6} & \text{if } x \in (-\infty, 0] \\ N_1(x) & \text{otherwise} \end{cases}$$

$$N_2^e(x) = \begin{cases} \frac{2}{3} & \text{if } x \in (-\infty, 0] \\ N_2(x) & \text{otherwise} \end{cases}$$

$$N_3^e(x) = \begin{cases} \frac{x}{2} + \frac{1}{6} & \text{if } x \in (-\infty, 0] \\ N_3(x) & \text{otherwise} \end{cases}$$

$$N_4^e(x) = \begin{cases} 0 & \text{if } x \in (-\infty, 0] \\ N_4(x) & \text{otherwise} \end{cases}$$

The extended basis for the rightmost boundary are obtained by symmetry. Figure 9 illustrates the resulting extended basis functions. The twodimensional counterparts of these newly defined extended basis functions are obtained using the tensor-product.

The proposed extension gives a remarkably good behavior to the extrapolating functions. See figures 10 and 11 for an illustration in 1D and 2D respectively. Besides, the fact that the basis functions form a partition of unity remains true ($\sum_{i=1}^{4} N_i^e(x) = 1, \forall x \in (-\infty, 0]$).

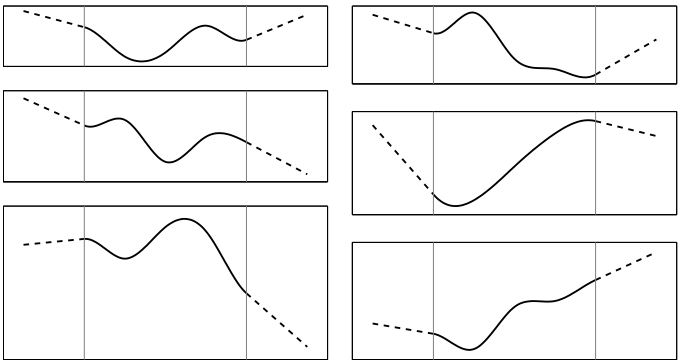


Figure 10: Examples of our extrapolating FFD in the monodimensional case. The extrapolating parts are represented with dashed lines.

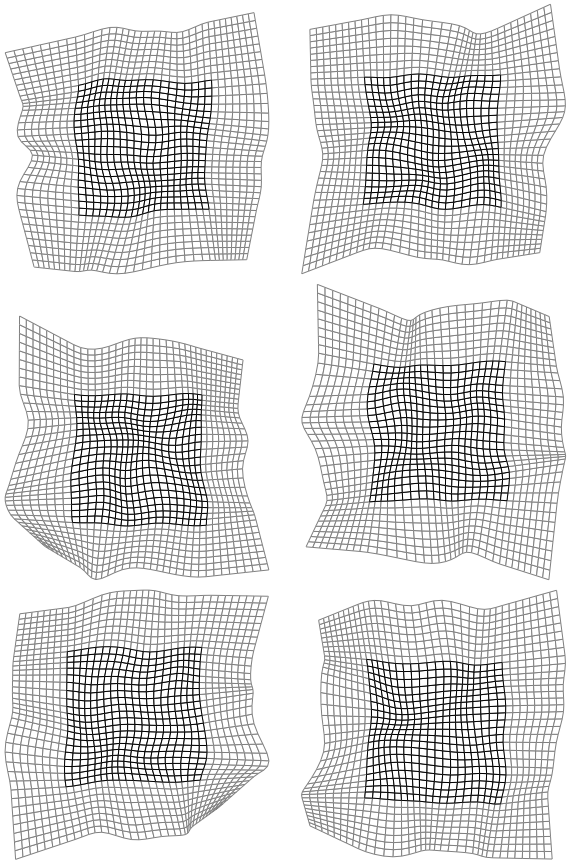


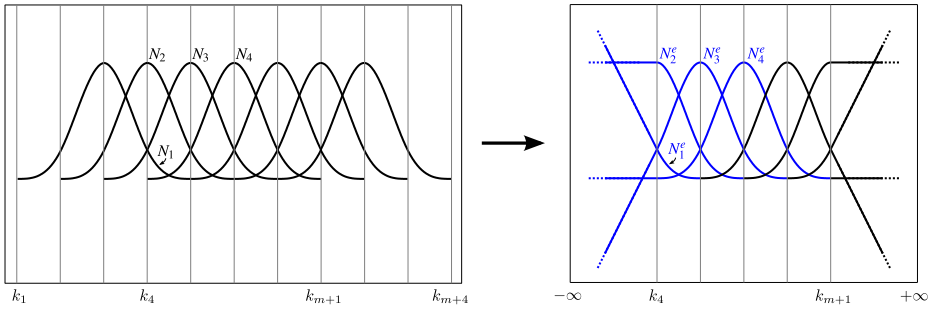
Figure 11: Examples of our extrapolating FFD warp. The dark part of the meshes represents the warp over its initial domain while the light part is extrapolated.

6 Experimental Results

6.1 Representational Similarity of the TPS and FFD Warps

This first set of experiments is designed to compare the TPS and the FFD warp. In the light of these experiments, we believe that a fair conclusion is that the TPS and the FFD warp have the same order of representational power. This motivates our choice to only consider the TPS warp in the other experiments of this article.

Fitting error from point correspondences. The experimental setup is as follows. We synthesize a set of point correspondences $\mathbf{q}_k \leftrightarrow \mathbf{a}_k$. The points $\mathbf{q}_k$ in the first image are taken as the nodes of a regular grid of size 11×11 over the domain $[-1, 1] \times [-1, 1]$. These points are randomly and independently



(a) (b)

Figure 9: (a) Standard basis functions. (b) Extended basis functions that allow one to extrapolate outside the natural domain $[k_4, k_{m+1}]$.

moved (with a given average magnitude γ) in order to build the corresponding points $\mathbf{a}_k$ in the second image. A TPS and an FFD warp are then estimated from the point correspondences. The initial data points $\mathbf{a}_k$ and the warped ones are compared using the *fitting error* defined by:

$$\frac{1}{n}\sum_{i=1}^{n} d\left(\mathbf{a}_i, \mathcal{W}(\mathbf{q}_i)\right), \tag{45}$$

where $\mathcal{W}$ represents the estimated warp (either TPS or FFD) and d the euclidean distance. Note that the fitting error is expressed in the same unit as the points of the second image. The results are shown in figure 12 for different displacement magnitudes. The reported values are obtained as the average over 500 trials. The fitting errors and the displacement magnitudes are expressed in percentage of the domain size. We can see that the curves in figure 12 are almost identical. It means that none of the two considered warps prevails the other one ; they can model equally the point correspondences. Note also that the fitting error collapses to zero when the number of centers is the same as the number of point correspondences.

Direct comparison of the warps. This second experiment differs from the first one in the sense that the point correspondences are not generated randomly. First, a TPS warp is generated with randomly determined driving features $\mathbf{v}_\mathsf{T}$. The centers of this TPS warp are taken on an $n \times n$ regular grid over the domain $\Omega = [-1, 1] \times [-1, 1]$. The generated driving features $\mathbf{v}_\mathsf{T}$ are produced by moving the centers around their initial location with an average magnitude γ. Second, a set of point correspondences is synthesized by sampling the TPS warp: $\mathbf{q}_k \leftrightarrow \mathbf{a}_k = \mathcal{W}_\mathsf{T}(\mathbf{q}_k; \mathbf{v}_\mathsf{T})$. Third, the features $\mathbf{v}_\mathsf{F}$ of an FFD warp are determined from the previously generated point correspondences. Finally, the TPS and the FFD warps are compared using the following measure:

$$\iint_\Omega \|\mathcal{W}_\mathsf{T}(\mathbf{q}; \mathbf{v}_\mathsf{T}) - \mathcal{W}_\mathsf{F}(\mathbf{q}; \mathbf{v}_\mathsf{F})\| \, \mathrm{d}\mathbf{q}. \tag{46}$$

The unit of this error measure is the same as the one of the points in the second image. The results are presented in figure 13. Figure 14 are the results obtained with the same experimental setup except that the roles of the TPS and the FFD warps have been switched. The reported errors are computed by averaging over 200 trials. These errors are expressed in percentage of the domain size.

Figure 13 tells us that given a TPS warp, it is possible to closely approximate the same deformations with an FFD warp. Conversely, the deformations induced by an FFD warp can also be closely approximated by a TPS warp (figure 14).

Comparison of the TPS and FFD warps induced by the same driving features. Since the parameters in the Feature-Driven framework are meaningful to the warp (*i.e.* they are interpolated by the warps), we can compare the TPS and the FFD warps with the same set of driving features. In this experiment, we randomly generate some driving features with their associated centers. We then compare the TPS and the FFD warps resulting from these features using the same measure as in the previous experiment. Figure 15 shows the results for different magnitude of transformation. The reported numbers are obtained as the average over 200 trials.

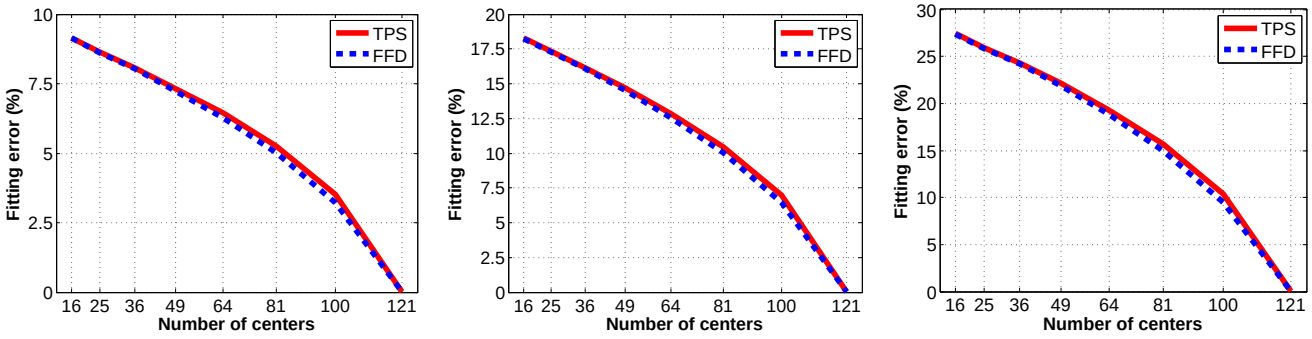
The values of the errors and the displacement magnitudes in figure 15 are expressed in percentage of the domain size. We can see from figure 15 that TPS and FFD warps induced from the same set of features are close to each other.

6.2 Comparison of Registration Algorithms

In this second set of experiments, we compare four algorithms in terms of convergence frequency, accuracy and convergence rate:

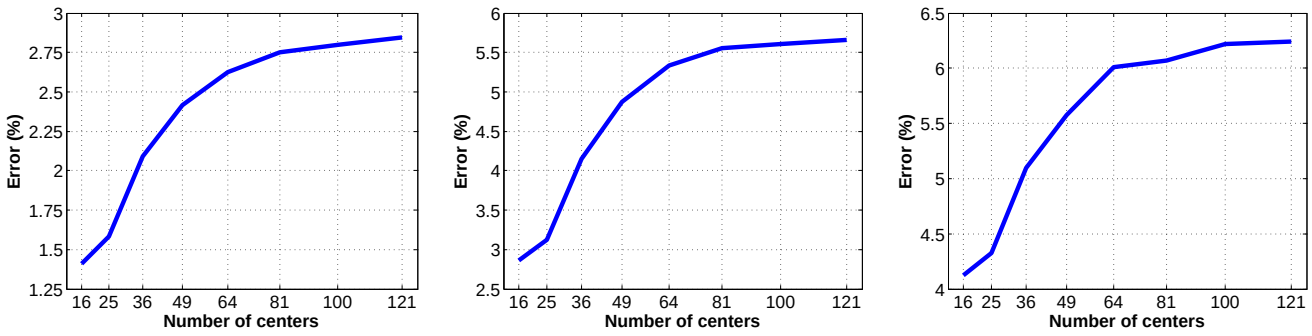
- Two classical algorithms:
  - **FA-GN** : the Forward Additive Gauss-Newton approach used by (Bartoli and Zisserman, 2004; Lim and Yang, 2005) and described in §2.1.
  - **FA-ESM** : the Efficient Second-order approximation of the cost function proposed by (Benhimane and Malis, 2004) and reviewed in §2.1 with an additive update of the parameters[3].
- Two algorithms we propose:
  - **IC-GN** : the Feature-Driven Inverse Compositional registration of §3 with Gauss-Newton as local registration engine as described in §4.1.

[3]The original ESM algorithm uses a compositional update. The Feature-Driven framework naturally extends it to non-rigid warps.



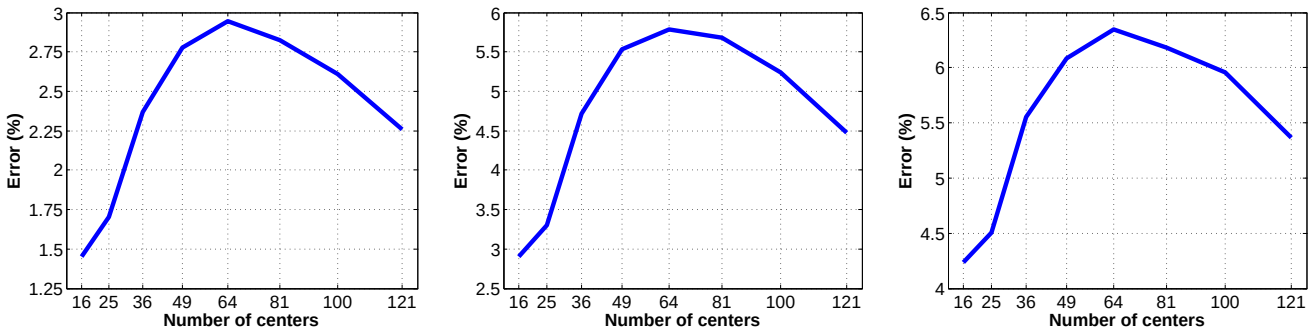
(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

Figure 12: Fitting error between the synthesized data points and the points warped with the estimated TPS and FFD warps (γ is the average displacement magnitude.)



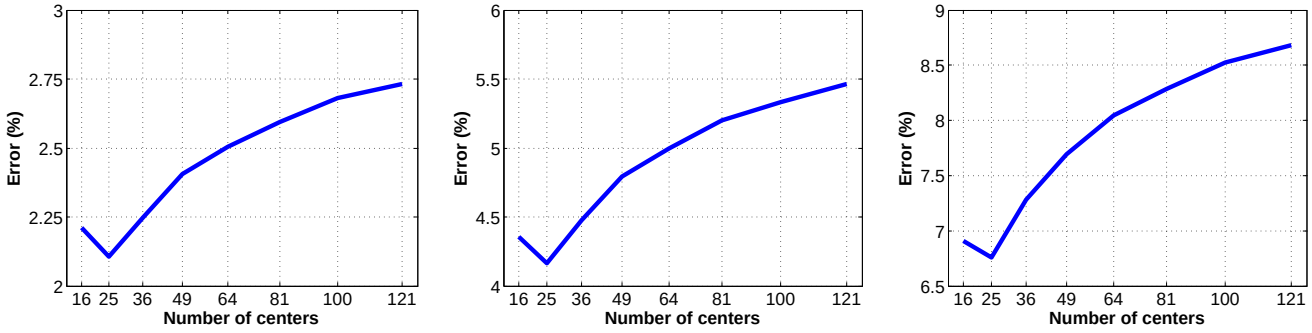
(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

Figure 13: Error between the FFD and the TPS warps. The FFD warp is estimated from point correspondences that comes from the TPS warp (γ is the displacement magnitude).



(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

Figure 14: Error between the TPS and the FFD warps. The TPS warp is estimated from point correspondences that come from the FFD warp (γ is the displacement magnitude).



(a) $\gamma = 10\%$ (b) $\gamma = 20\%$ (c) $\gamma = 30\%$

Figure 15: Comparison of the TPS and of the FFD warps resulting from a common set of driving features (γ is the displacement magnitude).

– **FC-LE** : the Feature-Driven Forward Compositional registration of §3, with local registration achieved through learning as described in §4.2.

6.2.1 Simulated Data

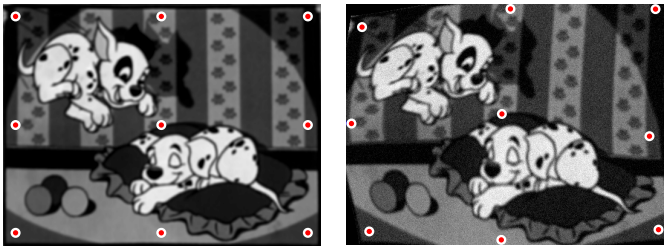
In order to assess algorithms in different controlled conditions, we synthesized images from a texture image. The driving features are placed on a 3×3 grid, randomly perturbed with magnitude r. We add Gaussian noise, with variance $\sigma\%$ of the maximum greylevel value, to the warped image. An example of such generated data is shown in figure 16. We vary each of these parameters independently, using the following default values: $r = 2$ pixels and $\sigma = 1\%$. The estimated warps are scored by the mean Euclidean distance between the driving features which generated the warped image, and the estimated ones. Convergence to the right solution is declared if this score is lower than one pixel. The characteristics we measured are:

- **Convergence frequency.** This is the percentage of convergence to the right solution.
- **Accuracy.** This is the mean residual error over the trials for which an algorithm converged.
- **Convergence rate.** This is defined by the number of iterations required to converge.

The results are averages over 500 trials. Note that, in this section, the elements in the legend of each figure are ordered according to their performance (from best to worst).

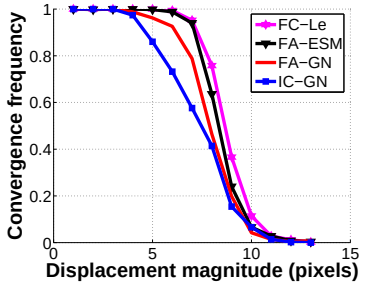
Convergence frequency. The results are shown in figure 17. FC-LE has the largest convergence basin closely followed by FA-ESM. IC-GN has the smallest convergence basin. At a displacement magnitude of 8 pixels, the convergence frequency of FC-LE is around 75% whereas it is near only 40% for FA-GN and IC-GN. FA-GN has the worst performances against noise. The other algorithms are almost unaffected by noise.

Accuracy. The results are shown in figure 18. The four algorithms are equivalent against the displacement magnitude. Concerning the amplitude of the noise, IC-GN and FC-LE are equivalent while FA-ESM is slightly worse and FA-GN clearly

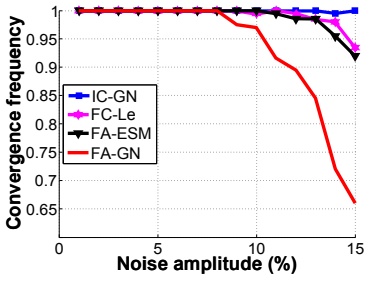


(a) (b)

Figure 16: An example of simulated data. (a) The texture image. (b) The warped image with gaussian noise added (using $\sigma = 5\%$ and $r = 8$ pixels).



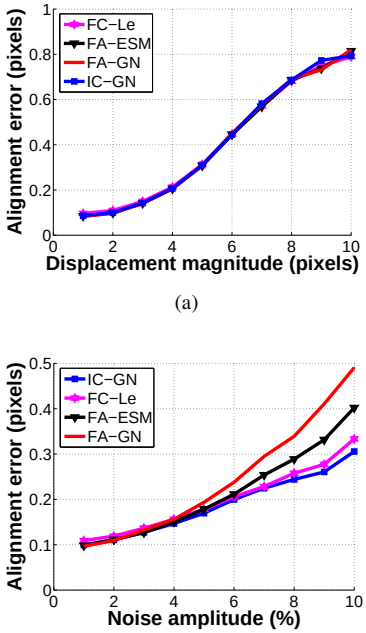
(a)



(b)

Figure 17: Comparison of the four algorithms in terms of convergence frequency against displacement magnitude (a) and noise amplitude (b).

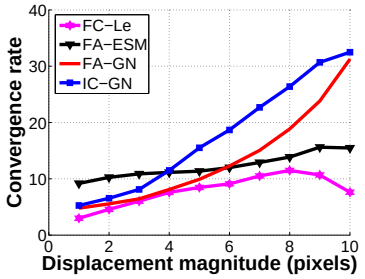
worse. For example, at 6% noise, the registration errors of IC-GN and FC-LE are around 0.2 pixels, FA-ESM is at about 0.25 pixels and FA-GN at 0.35 pixels.



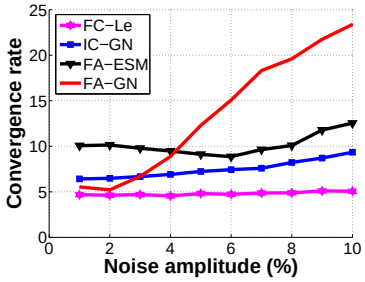
(a)

(b)

Figure 18: Comparison of the four algorithms in terms of accuracy against displacement magnitude (a) and noise amplitude (b).



(a)



(b)

Figure 19: Comparison of the four algorithms in terms of convergence rate against displacement magnitude (a) and noise amplitude (b).

Convergence rate. The results are shown in figure 19. The convergence rate of FC-LE and FA-ESM are almost constant against both displacement and noise amplitude. However FC-LE does better, with a convergence rate kept below 10 iterations. FA-GN and IC-GN are efficient for small displacements, *i.e.* below 5 pixels. The convergence rate increases dramatically beyond this value for both of them. FA-GN is also inefficient for noise amplitude over 4%. This is explained by the fact that the FA-GN Jacobian matrix depends mainly on the current image gradient, onto which the noise is added.

Discussion. This set of experiments on synthetic data shows that the proposed algorithms, *i.e.* FC-LE and IC-GN, are always the best ones in the presence of noise. FC-LE also obtains the best performances against the displacement magnitude. However, the standard algorithms, *i.e.* FA-ESM and FA-GN, performs better than IC-GN for important displacement magnitudes.

6.2.2 Real Data

The four above described algorithms have been compared on several image sequences. We show results for four such videos[4]. We measured the average and maximum intensity RMS along the sequence, the total number of iterations and the computational time (expressed in seconds). The RMS is expressed in pixel value unit. Note that the RMS is computed on the pixels of interest, *i.e.* the pixels actually used in the registration algorithms themselves. All algorithms have been implemented in Matlab. In order to illustrate the registration, we defined a mesh on the texture image and transferred it to all the other frames. Note that these meshes are different from the estimated driving features. The registration differences between the four algorithms are generally visually indistinguishable when they converge.

The first T-shirt sequence. This sequence has 400 frames. The displacement magnitude between the frames may be important. The driving features of the warp are placed on a 3×3 grid. Results are given in table 3 and registration for the FC-LE algorithm shown in figure 20. FC-LE performs well on this sequence. It is the fastest and the most accurate. FA-GN, FA-ESM and IC-GN are quite equivalent in terms of alignment accuracy. FA-GN needs a lot of iterations, making it 5 times slower than FC-LE.

The paper sequence. This sequence has 350 frames. The driving features of the warp are placed on a 4×4 grid. The results are given in table 4 and registration for the FC-LE algorithm shown in figure 21. IC-GN diverges when the deformation seems to be the most important. The other algorithms have similar alignment performances, FA-GN being slightly better. FC-LE is however 3 times faster than the other algorithms.

The rug sequence. This short sequence has 42 frames. The displacement magnitude is high. The driving features of the warp are placed on a 5×5 grid. The results are given in table 5 and registration for the FC-LE algorithm shown in figure 22.

[4]These videos can be downloaded at `http://laic.u-clermont1.fr/~brunet/ijcv`.

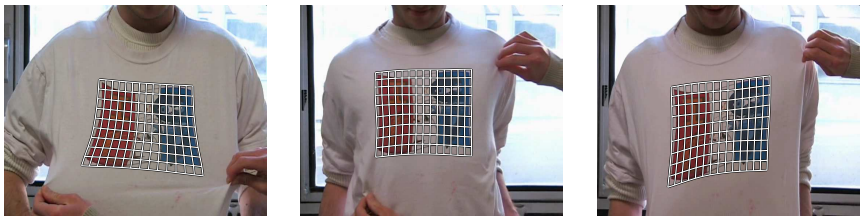


Figure 20: Registration results for FC-LE on the first T-shirt sequence.

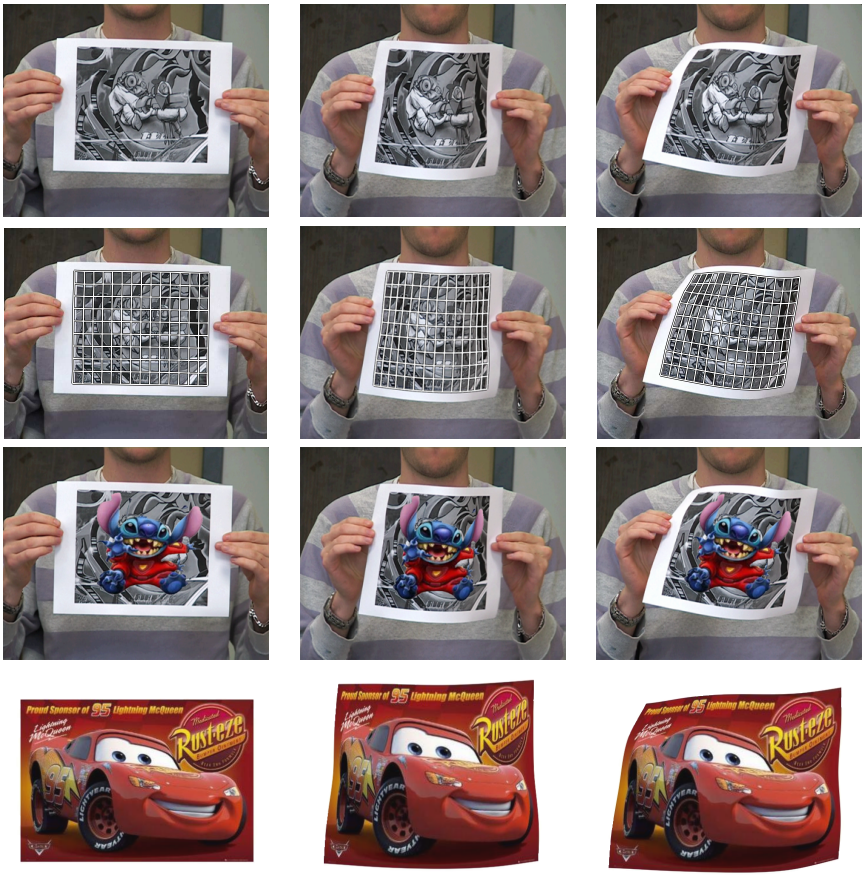


Figure 21: First row: the paper sequence. Second row: registration results for FC-LE. Third row: surface augmentation with the Walt Disney character *Stitch*. Fourth row: deformations are captured and applied on a poster of the Walt Disney movie *Cars*.

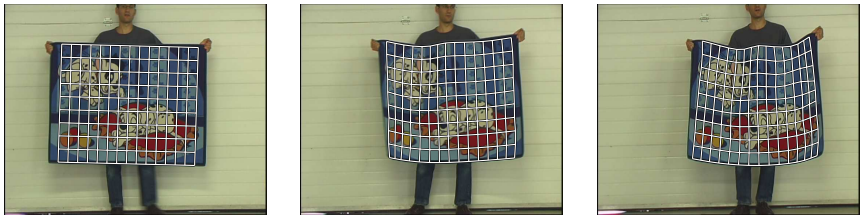


Figure 22: Registration results for FC-LE on the rug sequence.

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	8.70/13.67	9057	2083/5.2
FA-ESM	9.23/14.77	3658	877/2.2
IC-GN	9.69/15.82	6231	436/1.1
FC-LE	**6.66/12.87**	**3309**	**380/0.95**

Table 3: Results for the first T-shirt sequence. Bold indicates best performances.

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	**8.98/17.57**	2422	532/1.5
FA-ESM	10.22/20.49	2473	560/1.6
FC-LE	9.44/19.4	**1330**	**176/0.5**

Table 4: Results for the paper sequence. The IC-GN algorithm diverges on this sequence. Bold indicates best performances.

As for the paper sequence, IC-GN diverges. FA-GN and FA-ESM give the most accurate alignment, FC-LE being slightly worse. On the other hand it is 7 times faster.

The second T-shirt sequence. This sequence has 623 frames. Deformations are moderate, but there are strong global illumination variations along the sequence. The driving features of the warp are placed on a 3×3 grid. The results are given in table 6 and registration results for the FC-LE algorithm shown in figure 23. The registration is well achieved by the four algorithms. The small residual error shows that the global illumination changes are correctly compensated. FC-LE and IC-GN are respectively 4 and 2 times faster than the classical algorithms.

Discussion. FA-GN is the most accurate algorithm. It is however inefficient, especially for important displacements. FA-ESM has almost similar performances compared to the FA-GN while being slightly more efficient. IC-GN is efficient, but looses effectiveness for high displacements. As for the experiments with synthetic data, FC-LE has the best behavior: it is similar to FA-GN for accuracy while being 5 times faster on average and is equivalent or better than IC-GN and FA-ESM in terms of alignment accuracy, computational cost and has a larger convergence basin.

7 Conclusions

We addressed an important issue for the problem of non-rigid registration. We proposed the Feature-Driven framework, relaxing the groupwise requirement for using efficient compositional algorithms such Inverse Compositional and Learning-Based algorithms. We also explained in details the Feature-Driven parameterization for the TPS and the FFD warps. Experiments show that Feature-Driven algorithms are more efficient compared to classical ones with additive update of the parameters. Overall, the best algorithm is the combination of the Feature-Driven framework, the Forward Compositional update of the parameters and the local registration based on Learning. The proposed algorithms make foreseeable accurate real-time surface registration.

Acknowledgements. We would like to thank Selim Benhimane for his useful advice, in particular for proposing the Gaussian Mixture Model in order to select the weights of the interaction matrices in the piecewise linear model used in the learning-based local registration.

References

Agarwal A, Triggs B (2006) Recovering 3D human pose from monocular images. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(1)

Baker S, Matthews I (2004) Lucas-Kanade 20 years on: A unifying framework. International Journal of Computer Vision 56(3):221–255

Bartoli A, Zisserman A (2004) Direct estimation of non-rigid registrations. In: Proceedings of the British Machine Vision Conference

Benhimane S, Malis E (2004) Real-time image-based tracking of planes using efficient second-order minimization. In: Proceedings of the International Conference on Intelligent Robots and Systems

Bookstein FL (1989) Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(6):567–585

Bregler C, Hertzmann A, Biermann H (2000) Recovering non-rigid 3D shape from image streams. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition

Charpiat G, Faugeras O, Keriven R (2005) Image statistics based on diffeomorphic matching. In: Proceedings of the International Conference on Computer Vision

Cootes TF, Edwards GJ, Taylor CJ (1998) Active appearance models. In: Proceedings of the European Conference on Computer Vision

Cootes TF, Marsland S, Twining CJ, Smith K, Taylor CJ (2004) Groupwise diffeomorphic non-rigid registration for automatic model building. In: Proceedings of the European Conference on Computer Vision

De Boor C (2001) A Practical Guide to Splines (Revised Edition). Springer

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	**5.64/8.21**	538	118/2.8
FA-ESM	5.79/8.63	477	109/2.6
FC-LE	6.45/9.80	**149**	**17.13/0.4**

Table 5: Results for the rug sequence. The IC-GN algorithm diverges on this sequence. Bold indicates best performances.

	Mean/max RMS	# Iteration	Total/mean time
FA-GN	**4.51/7.42**	3408	785/1.25
FA-ESM	4.53/7.49	3268	788/1.25
IC-GN	4.87/7.61	4407	381/0.61
FC-LE	4.61/7.70	**1757**	**247/0.39**

Table 6: Results for the second T-shirt sequence. Bold indicates best performances.

Fornefett M, Rohr K, Stiehl H (1999) Elastic registration of medical images using radial basis functions with compact support. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition

Gay-Bellile V, Perriollat M, Bartoli A, Sayd P (2006) Image registration by combining thin-plate splines with a 3D morphable model. In: Proceedings of the International Conference on Image Processing

Gay-Bellile V, Bartoli A, Sayd P (2007) Feature-driven direct non-rigid image registration. In: Proceedings of the British Machine Vision Conference

Georgel P, Benhimane S, Nassir N (2008) A unified approach combining photometric and geometric information for pose estimation. In: Proceedings of the British Machine Vision Conference

Haber E, Modersitzki J (2006) Intensity gradient based registration and fusion of multi-modal images. In: Berlin S (ed) Proceedings of the Medical Image Computing and Computer-Assisted Intervention, Lecture Notes in Computer Science, vol 4191, pp 726–733

Johnson HJ, Christensen GE (2002) Consistent landmark and intensity-based image registration. IEEE Transactions on Medical Imaging 21(5):450–461

Joshi S, Miller MI (2000) Landmark matching via large deformation diffeomorphisms. IEEE Transactions on Image Processing 9:1357–1370

Jurie F, Dhome M (2002) Hyperplane approximation for template matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):996–1000

Lim J, Yang MH (2005) A direct method for non-rigid motion with thin-plate spline. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition

Little JA, Hill DLG, Hawkes DJ (1997) Deformations incorporating rigid structures. Computer Vision and Image Understanding 66(2):223–232

Matthews I, Baker S (2004) Active appearance models revisited. International Journal of Computer Vision 60(2):135–164

Meyer CR, Boes JL, Kim B, Bland PH, Zasadny KR, Kison PV, Koral K, Frey KA, Wahl RL (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(3):195–206

Pilet J, Lepetit V, Fua P (2005) Real-time non-rigid surface detection. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition

Pizarro D, Bartoli A (2007) Shadow resistant direct image registration. In: Proceedings of the Scandinavian Conference on Image Analysis, pp 928–937

Pluim JPW, Maintz JBA, Viergever MA (2003) Mutual-information-based registration of medical images: a survey. IEEE Transactions on Medical Imaging 22:986–1004

Romdhani S, Vetter T (2003) Efficient, robust and accurate fitting of a 3D morphable model. In: Proceedings of the International Conference on Computer Vision

Rueckert D, Sonoda LI, Hayes C, Hill DL, Leach MO, Hawkes DJ (1999) Nonrigid registration using free-form deformations: application to breast MR images. IEEE Transactions on Medical Imaging 18(8):712–721

Szeliski R (2006) Image alignment and stitching: A tutorial. Foundations and Trends in Computer Graphics and Vision 2:1–104

Torr PHS, Zisserman A (1999) Feature based methods for structure and motion estimation. In: Workshop on Vision Algorithms: Theory and Practice

Wahba G (1990) Spline models for observational data, CBMS-NSF Regional Conference Series in Applied Mathematics, vol 59. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA

Figure 23: Registration results for FC-LE on the second T-shirt sequence.

A Learning-based Registration with a Piecewise Linear Model

A.1 Framework

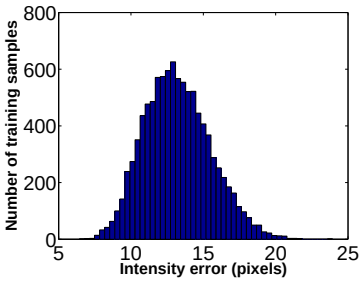
Learning-based registration algorithms such as (Cootes et al, 1998; Jurie and Dhome, 2002) use a linear model *i.e.* a single interaction matrix. It has several drawbacks: if this interaction matrix covers a large domain of deformation magnitudes, registration accuracy is spoiled. On the other hand, if the matrix is learned for small deformations only, the convergence basin is dramatically reduced. We propose to learn a series $\mathsf{F}_1, \ldots, \mathsf{F}_\kappa$ of interaction matrices, each of them covering a different range of displacement magnitudes. This forms a piecewise linear approximation to the true relationship. Experiments show that using a piecewise linear relationship solves these issues. We propose several ways to combine the interaction matrices $\{\mathsf{F}_i\}_{i=1}^{\kappa}$, that yield different piecewise linear relationships.

Trivial combination. One possibility is to apply all the matrices in turn (algorithm LOOP). The interaction matrix F_1 is first applied until convergence. Then, the other matrices $\{\mathsf{F}_i\}_{i=2}^{\kappa}$ are used one after the other. The last matrix F_κ, learned on the smallest displacement, ensures accuracy. The drawback is that all the matrices are used even for small displacements. It yields a dramatically high number of iterations to converge. Another approach is to try all the linear relationships at each iteration. The one resulting in the smallest residual error is kept (algorithm BEST). These piecewise linear relationships appear not to be the most discerning choice since they are not efficient. In fact, LOOP implies a large convergence rate whereas BEST yields high computational cost per iteration. They are less efficient when the number of interaction matrices κ increases.

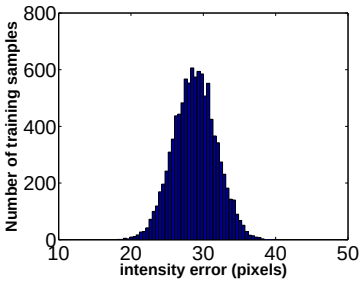
Statistical map selection. Our goal is to select the most appropriate interaction matrix at each iteration (algorithm PROB). Each of those indeed has a specific domain of validity in the displacement magnitude. This can unfortunately not be determined prior to image registration. We thus propose to learn a relationship between the intensity error magnitude and displacement magnitude intervals. We express this relationship in terms of probabilities. The intensity error magnitude for a residual vector $\mathbf{d}$ is defined as its RMS: $e(\mathbf{d}) = \text{rms}(\mathbf{d})$. Experimentally, we observed that $P(\mathsf{F}_i|e(\mathbf{d}))$ closely follows a Gaussian distribution, see figure 24.

Given the current intensity error $e(\mathbf{d})$, finding the most appropriate interaction matrix F_s is simply achieved by solving:

$$s = \arg\max_t P(\mathsf{F}_t|e(\mathbf{d})). \quad (47)$$



(a)



(b)

Figure 24: Distribution of the intensity error magnitude for two different perturbation intervals: (a) $[2 \ldots 5]$ and (b) $[7 \ldots 13]$ pixels, respectively.

Mixture models. We use an interaction matrix given by:

$$\mathsf{F} \leftarrow \sum_{i=1}^{\kappa} a_i \mathsf{F}_i \qquad \text{with} \sum_{i=1}^{\kappa} a_i = 1, \tag{48}$$

where $\{a_i\}_{i=1}^{\kappa}$ are the mixture proportions.

We compare two mixture models with different probability distributions:

- a Gaussian Mixture Model (GMM): $a_i = \frac{P(\mathsf{F}_i|e(\mathbf{d}))}{\sum_{k=1}^{\kappa} P(\mathsf{F}_k|e(\mathbf{d}))}$, where $P(\mathsf{F}_i|e(\mathbf{d}))$ follows a Gaussian distribution;
- a Constant Mixture Model (CMM): $a_i = \frac{1}{\kappa}$.

For all the piecewise linear relationships described above[5], the interaction matrix F_κ is applied for the last two iterations. This additional step ensures accuracy.

A.2 Experimental Results

We use a simple homographic warp to guarantee a fair comparison[6] between the five piecewise linear relationships. The setup described in §6.2.1 is used.

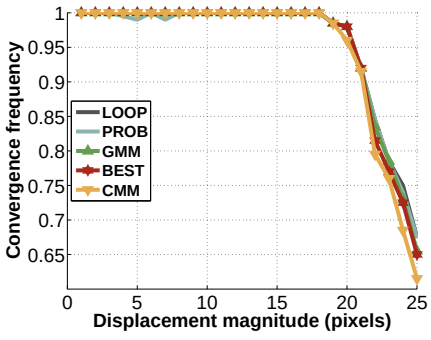
Convergence frequency. The results are shown in figure 25. The five piecewise linear relationships have similar convergence basins. Their convergence frequency are over 95% and around 65% at a displacement magnitude of 20 pixels and 25 pixels respectively. CMM seems to have a slightly thiner convergence basin. GMM and PROB are quite sensitive to noise. It corrupts the probability distributions learned off-line. GMM is slightly better than PROB: at a noise magnitude of 7% the convergence frequency of PROB is only 80% whereas GMM always converges. CMM, BEST and LOOP are insensitive to noise.

Accuracy. The results against displacement magnitude are similar for the five piecewise linear relationships. The associated graph is thus not shown. The residual error is around 0.025 pixels for all tested magnitudes. GMM and PROB are less accurate against noise beyond an amplitude of 7% and 9% respectively than the other piecewise linear relationships. This is illustrated in figure 26(b). Their residual errors are approximately 0.7 pixels against 0.3 pixels for CMM, BEST and LOOP (*i.e.*, a 2.5 ratio).

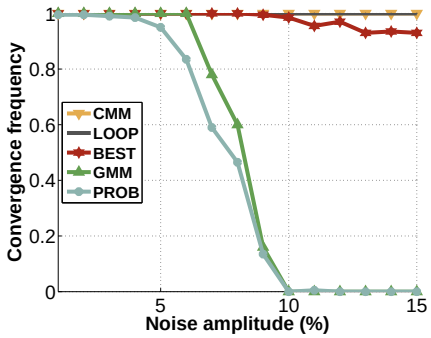
Convergence rate. The results are shown in figure 26(a). BEST and LOOP are clearly not efficient against displacement magnitude: LOOP needs at least 30 iterations to converge while BEST requires 10 iterations with high computational cost. CMM, GMM and PROB do better with a convergence rate kept below 10. Convergence rates for the five piecewise linear relationships are similar against noise magnitude. The associated graph is thus not shown.

[5]Except LOOP which naturally includes this additional step.

[6]The compositional update of the warp is not approximated since an homography belongs to a group.

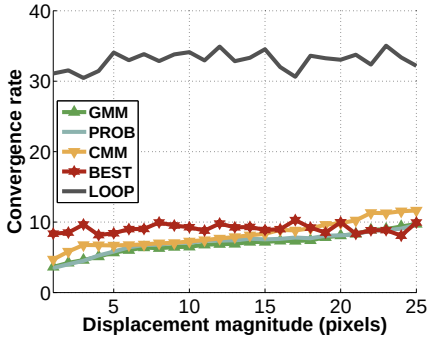


(a)

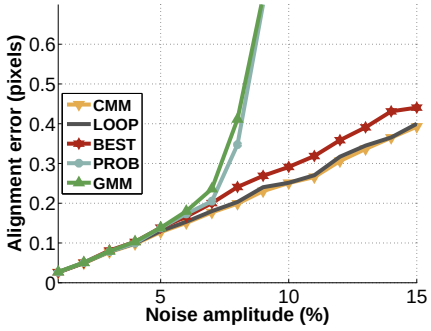


(b)

Figure 25: Comparison of the five piecewise linear relationships in terms of convergence frequency against displacement magnitude (a) and noise amplitude (b). On graph (b), CMM and LOOP are indistinguishable.



(a)



(b)

Figure 26: (a) Comparison of the five piecewise linear relationships in terms of convergence rate against displacement magnitude. (b) Comparison of the five piecewise linear relationships in terms of accuracy against noise amplitude.

Discussion. Overall, CMM is the best piecewise relationship. It is much more efficient than LOOP and BEST and unaffected by noise contrary to CMM and PROB. Its convergence basin is only slightly smaller than those induced by the other relationships. We use the CMM piecewise linear relationship in the experiments of §6.2.