



HAL
open science

Improvements of the Inverse Compositional Algorithm for Parametric Motion Estimation

Thibaud Briand, Gabriele Facciolo, Javier Sanchez

► **To cite this version:**

Thibaud Briand, Gabriele Facciolo, Javier Sanchez. Improvements of the Inverse Compositional Algorithm for Parametric Motion Estimation. *Image Processing On Line*, 2018, pp.2018 - 2029. 10.5201/ipol.2018.222 . hal-01959765

HAL Id: hal-01959765

<https://hal.science/hal-01959765>

Submitted on 19 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Published in *Image Processing On Line* on 2018–12–18.
 Submitted on 2018–02–26, accepted on 2018–11–27.
 ISSN 2105–1232 © 2018 IPOL & the authors [CC-BY-NC-SA](#)
 This article is available online with supplementary materials,
 software, datasets and online demo at
<https://doi.org/10.5201/ipol.2018.222>

Improvements of the Inverse Compositional Algorithm for Parametric Motion Estimation

Thibaud Briand^{1,2}, Gabriele Facciolo², and Javier Sánchez³

¹ Université Paris-Est, LIGM (UMR CNRS 8049), ENPC, F-77455 Marne-la-Vallée, France
 (thibaud.briand@enpc.fr)

² Université Paris-Saclay, CMLA (UMR CNRS 8536), ENS Cachan, 94235 Cachan, France
 (facciolo@cmla.ens-cachan.fr)

³ CTIM, Department of Computer Science, University of Las Palmas de Gran Canaria, Spain
 (jsanchez@ulpgc.es)

Abstract

In this work, we propose several improvements of the inverse compositional algorithm for parametric registration. We propose an improved handling of boundary pixels, a different color handling and gradient estimation, and the possibility to skip scales in the multiscale coarse-to-fine scheme. In an experimental part, we analyze the influence of the modifications. The estimation accuracy is at least improved by a factor 1.3 while the computation time is at least reduced by a factor 2.2 for color images.

Source Code

The C++ source code, the code documentation, and the online demo are accessible at [the web page of this article](#)¹. Compilation and usage instruction are included in the `README.txt` file of the archive.

Keywords: inverse compositional algorithm; parametric motion estimation; boundary handling; gradient estimation

1 Introduction

Image alignment is one of the most widely used techniques in computer vision. The objective of parametric motion estimation methods is to find the global transformation that puts in correspondence the pixels of two images. An accurate and efficient estimation is important in problems such as optical flow estimation, object tracking, video stabilization, image stitching or 3D reconstruction. The task is difficult because it deals with problems like occlusions, noise, local brightness changes or spurious motions. Methods can be classified into intensity-based and feature-based. Intensity-based methods

¹<https://doi.org/10.5201/ipol.2018.222>

are usually faster but more sensitive to brightness changes and outliers. Feature-based methods are typically more sensitive to noise and motion blur, but allow to estimate stronger deformations [21].

In this article we focus on the inverse compositional algorithm for parametric motion estimation. First introduced in [2, 22], it is an improvement of the classical intensity-based method of Lucas-Kanade [11]. At each step of its iterative scheme it solves a minimization problem equivalent to Lucas-Kanade but more efficiently. It allows for precomputations and the gradient of the reference image is not interpolated. The inverse compositional algorithm was studied in more detail in [3]. With the extension to robust error functions in [1], it becomes less sensitive to outliers. Better precision for large motions can be obtained with a coarse-to-fine multiscale approach as in [17]. Using the implementation provided by [17], it can be observed that, for moderate deformations, this method is faster and more accurate than classical feature-based methods that rely on SIFT keypoints [9, 16] and the RANSAC algorithm [6].

We claim that the inverse compositional algorithm can be further improved and accelerated with a correct handling of the boundary values, a different color handling and gradient estimation, and by skipping scales in the multiscale coarse-to-fine scheme. The estimation accuracy is at least improved by a factor 1.3 while the computation time is at least reduced by a factor 2.2 for color images. In this article, we detail the inverse compositional algorithm and analyze the influence of the proposed modifications. An implementation of the modified algorithm based on the one of [17] is also provided.

The rest of the paper is organized as follows. First, we detail the inverse compositional algorithm in Section 2 as it is presented in [17], i.e., with the use of robust error functions and a multiscale approach. In Section 3, we present the proposed modifications that lead to the modified inverse compositional algorithm. In the experimental part (Section 4) we study and discuss the influence of the modifications, and we finish with conclusions in Section 5.

2 The Inverse Compositional Algorithm for Parametric Registration

Let M, N, C be three positive integers. Define the spatial domain $\Omega = \Omega_{M,N} = \{0, \dots, M - 1\} \times \{0, \dots, N - 1\}$. Let I_1 and I_2 be two images of size $M \times N$ with C channels (e.g. $C = 1$ for grayscale images and $C = 3$ for color images). The channels are handled so that, for $\mathbf{x} \in \Omega$, $I_1(\mathbf{x}) = (I_1^{(1)}(\mathbf{x}), \dots, I_1^{(C)}(\mathbf{x}))^T \in \mathbb{R}^C$ is a vector of length C .

The motion between the two images is assumed to be representable by a parametric motion model. Denote by $\Psi(\cdot; \mathbf{p}) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ the transformation parametrized by $\mathbf{p} \in \mathbb{R}^n$. The parametric motion estimation problem is to find a motion parameter \mathbf{p}^* such that

$$\forall \mathbf{x} \in \Omega, \quad I_1(\mathbf{x}) \simeq I_2(\Psi(\mathbf{x}; \mathbf{p}^*)). \tag{1}$$

There is in practice no equality in (1) because, for instance, images contain noise and occlusions may occur. In addition, the motion model in general only approximates the real motion.

Additional hypotheses. In order to apply the inverse compositional algorithm, additional hypotheses are made on the set of transformations $\{\Psi(\cdot; \mathbf{p}), \mathbf{p} \in \mathbb{R}^n\}$ and its parametrization.

1. The motion parameter $\mathbf{p} = 0$ corresponds to the identity transformation, i.e. $\Psi(\mathbf{x}; 0) = \mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^2$.
2. The set of transformations has a group structure under composition of functions.
3. For all $\mathbf{x} \in \mathbb{R}^2$, the function $\mathbf{p} \in \mathbb{R}^n \mapsto \Psi(\mathbf{x}; \mathbf{p})$ is differentiable at $\mathbf{p} = 0$.

Transform	n	Parameters - \mathbf{p}	Matrix - $H(\mathbf{p})$	Jacobian - $J(x, y)$
Translation	2	(t_x, t_y)	$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Euclidean	3	(t_x, t_y, θ)	$\begin{pmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & -y \\ 0 & 1 & x \end{pmatrix}$
Similarity	4	(t_x, t_y, a, b)	$\begin{pmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{pmatrix}$
Affinity	6	$(t_x, t_y, a_{11}, a_{12}, a_{21}, a_{22})$	$\begin{pmatrix} 1+a_{11} & a_{12} & t_x \\ a_{21} & 1+a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{pmatrix}$
Homography	8	$(h_{11}, h_{12}, h_{13}, \dots, h_{32})$	$\begin{pmatrix} 1+h_{11} & h_{12} & h_{13} \\ h_{21} & 1+h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$	$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x^2 & -xy \\ 0 & 0 & 0 & x & y & 1 & -xy & -y^2 \end{pmatrix}$

Table 1: Planar transformations in their homogeneous coordinates and their Jacobians.


 Figure 1: Example of reference image I_1 (left) and warped image I_2 (right). These are color images of size 512×512 (i.e. $M = N = 512$ and $L = 3$) linked by a homography. The reference image is generated from the warped image by bicubic interpolation where outside pixel values are set to 0 (black region).

Table 1 lists the typical planar transformations and provides examples of parametrization. An example of images related by an homographic transformation is shown in Figure 1.

The inverse compositional algorithm tries to solve the parametric motion problem with an iterative scheme. Firstly, we start by introducing the mathematical construction on which the algorithm relies on. Secondly, we detail the inverse compositional algorithm along with all its parameters. Then, we discuss the influence of the error function. Finally, we present the coarse-to-fine multiscale approach.

2.1 Mathematical Construction

A good candidate for the parameter \mathbf{p}^* in (1) is a minimizer of the energy

$$\mathbf{p} \in \mathbb{R}^n \mapsto E_0(\mathbf{p}) = \sum_{\mathbf{x} \in \Omega} \rho (\|I_2(\Psi(\mathbf{x}; \mathbf{p})) - I_1(\mathbf{x})\|^2), \quad (2)$$

where $\rho : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is an increasing and derivable function called the *error function* whose influence is discussed in Section 2.3. As there is in general no explicit expression for computing a minimizer, it is approximated using an iterative scheme.

At a given step $j \geq 1$, the idea of the inverse compositional algorithm is to refine the current estimated transformation $\Psi(\cdot; \mathbf{p}_{j-1})$ with an inverted incremental transformation (hence the name of the algorithm) $\Psi(\cdot; \Delta \mathbf{p}_j)^{-1}$, i.e.,

$$\Psi(\cdot; \mathbf{p}_j) = \Psi(\cdot; \mathbf{p}_{j-1}) \circ \Psi(\cdot; \Delta \mathbf{p}_j)^{-1}. \quad (3)$$

The ideal choice for the increment $\Delta \mathbf{p}_j$ would be a minimizer of the incremental energy

$$\Delta \mathbf{p} \in \mathbb{R}^n \mapsto E_1(\Delta \mathbf{p}; \mathbf{p}_{j-1}) = \sum_{\mathbf{x} \in \Omega} \rho(\|I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1})) - I_1(\Psi(\mathbf{x}; \Delta \mathbf{p}))\|^2), \quad (4)$$

as it would give $\mathbf{p}_j = \mathbf{p}^*$ with (3) under the assumption that (1) holds. But, as for the original energy E_0 , such a minimizer cannot be computed and can only be approximated. Therefore the energy E_1 is approximated as follows using two successive first order Taylor expansions. Let $\mathbf{x} \in \Omega$.

First approximation. A first order Taylor expansion of the function $\Delta \mathbf{p} \in \mathbb{R}^n \mapsto I_1(\Psi(\mathbf{x}; \Delta \mathbf{p}))$ around 0 gives

$$I_1(\Psi(\mathbf{x}; \Delta \mathbf{p})) \simeq I_1(\mathbf{x}) + \nabla I_1^T(\mathbf{x}) J(\mathbf{x}) \Delta \mathbf{p}, \quad (5)$$

where

$$J(\mathbf{x}) = \frac{\partial \Psi}{\partial \mathbf{p}}(\mathbf{x}; 0) \in \mathcal{M}_{2,n} \quad (6)$$

is the Jacobian matrix of the model at \mathbf{x} and $\nabla I_1(\mathbf{x}) = \left(\frac{\partial I_1}{\partial x}(\mathbf{x}), \frac{\partial I_1}{\partial y}(\mathbf{x}) \right)^T \in \mathcal{M}_{2,C}$ is the gradient of I_1 at \mathbf{x} . The Jacobian matrices J of some typical planar transformations can be found in Table 1. For $C > 1$, ∇I_1 actually corresponds to the transposed of the Jacobian matrix of I_1 but to simplify we keep the gradient notation. Let us set

$$G(\mathbf{x}) = \nabla I_1^T(\mathbf{x}) J(\mathbf{x}) \in \mathcal{M}_{L,n}, \quad (7)$$

and denote DI the difference image defined by

$$DI(\mathbf{x}) = DI(\mathbf{x}; \mathbf{p}_{j-1}) = I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1})) - I_1(\mathbf{x}) \in \mathbb{R}^C. \quad (8)$$

Using (5) in (4), we define the approximated incremental energy

$$\Delta \mathbf{p} \in \mathbb{R}^n \mapsto E_2(\Delta \mathbf{p}; \mathbf{p}_{j-1}) = \sum_{\mathbf{x} \in \Omega} \rho(\|DI(\mathbf{x}) - G(\mathbf{x}) \Delta \mathbf{p}\|^2). \quad (9)$$

Second approximation. A first order Taylor expansion of the function $t \in \mathbb{R} \mapsto \rho(\|DI(\mathbf{x})\|^2 + t)$ around 0 gives

$$\rho(\|DI(\mathbf{x})\|^2 + t) \simeq \rho(\|DI(\mathbf{x})\|^2) + \rho'(\|DI(\mathbf{x})\|^2) t. \quad (10)$$

Using the expansion

$$\|DI(\mathbf{x}) - G(\mathbf{x}) \Delta \mathbf{p}\|^2 = \|DI(\mathbf{x})\|^2 - 2\Delta \mathbf{p}^T G(\mathbf{x})^T DI(\mathbf{x}) + \Delta \mathbf{p}^T G(\mathbf{x})^T G(\mathbf{x}) \Delta \mathbf{p} \quad (11)$$

and $t = -2\Delta \mathbf{p}^T G(\mathbf{x})^T DI(\mathbf{x}) + \Delta \mathbf{p}^T G(\mathbf{x})^T G(\mathbf{x}) \Delta \mathbf{p}$ in (10), we obtain the approximation

$$\rho(\|DI(\mathbf{x}) - G(\mathbf{x}) \Delta \mathbf{p}\|^2) \simeq \rho(\|DI(\mathbf{x})\|^2) + \rho'(\|DI(\mathbf{x})\|^2) (-2\Delta \mathbf{p}^T G(\mathbf{x})^T DI(\mathbf{x}) + \Delta \mathbf{p}^T G(\mathbf{x})^T G(\mathbf{x}) \Delta \mathbf{p}). \quad (12)$$

To simplify we denote $\tilde{\rho}'(\mathbf{x}) = \tilde{\rho}'(\mathbf{x}; \mathbf{p}_{j-1}) = \rho'(\|DI(\mathbf{x})\|^2) \in \mathbb{R}^+$. We define the Hessian matrix $H \in \mathcal{M}_n$ and the vector $\mathbf{b} \in \mathbb{R}^n$ by

$$H = H(\mathbf{p}_{j-1}) = \sum_{\mathbf{x} \in \Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x}), \quad (13)$$

$$\mathbf{b} = \mathbf{b}(\mathbf{p}_{j-1}) = \sum_{\mathbf{x} \in \Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T DI(\mathbf{x}). \quad (14)$$

Using (12), we define a second approximated incremental energy

$$\Delta \mathbf{p} \in \mathbb{R}^n \mapsto E_3(\Delta \mathbf{p}; \mathbf{p}_{j-1}) = \sum_{\mathbf{x} \in \Omega} \rho(\|DI(\mathbf{x})\|^2) - 2\Delta \mathbf{p}^T \mathbf{b} + \Delta \mathbf{p}^T H \Delta \mathbf{p}. \quad (15)$$

Assuming that this quadratic form is non-degenerate (i.e. that the symmetric matrix H is positive definite), we define the increment $\Delta \mathbf{p}_j$ as its unique minimizer that is given by

$$\Delta \mathbf{p}_j = H^{-1} \mathbf{b}. \quad (16)$$

2.2 Algorithm

In practice, the gradient of I_1 is estimated using the central differences scheme, i.e. for $\mathbf{x} = (x, y) \in \Omega$,

$$\frac{\partial I_1}{\partial x}(\mathbf{x}) \simeq \frac{1}{2} (I_1(x+1, y) - I_1(x-1, y)), \quad (17)$$

$$\frac{\partial I_1}{\partial y}(\mathbf{x}) \simeq \frac{1}{2} (I_1(x, y+1) - I_1(x, y-1)). \quad (18)$$

The difference image DI defined in (8) requires the values $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1}))$, which are computed by bicubic interpolation [7]. Neumann boundary conditions are adopted for both, the gradient estimation, and the interpolation. As in [17], when $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1}))$ has to be evaluated outside of the domain $[0, M-1] \times [0, N-1]$ its values are arbitrarily set to 0. This may be useful for simulated images as the one shown in Figure 1 but it is not adapted to real data. In Section 3.2 we discuss the impact of this choice and propose a strategy to avoid introducing bias.

Given an initialization $\mathbf{p}_0 \in \mathbb{R}^n$, the \mathbf{p}_j 's can be computed using (3) and (16), and are expected to be close to a minimizer of the original energy E_0 after enough iterations. The iterations are stopped as soon as the increment $\Delta \mathbf{p}_j$ is small enough. More precisely, for a given threshold $\epsilon > 0$, the stopping criterion is

$$\|\Delta \mathbf{p}_j\| \leq \epsilon. \quad (19)$$

Because the sequence $(\mathbf{p}_j)_{j \in \mathbb{N}}$ is not guaranteed to converge, a maximum number of iterations j_{\max} is also set. When the error function ρ depends on a threshold parameter we adjust it during the iterations as explained in Section 2.3. The one-scale inverse compositional algorithm is shown in Algorithm 1.

This algorithm is an improvement of the Lucas-Kanade method [11] since at each step of the incremental refinement it solves an equivalent minimization problem but more efficiently [2]. As ∇I_1 , G , and $G^T G$ do not depend on \mathbf{p}_{j-1} they are precomputed before the incremental refinement. The Hessian H can also be precomputed for the L2 error function (see Section 2.3). Note that precomputing is memory greedy and may be replaced by in-place computations. For instance the precomputation of $G^T G$ requires to store $n^2 M N$ values. In addition, the gradient of the reference image is not interpolated during the incremental refinement.

Note that contrast change may deteriorate the algorithm performance since it is not taken into account in (1). It can be handled by equalizing the input image contrasts, for instance using the Midway Image Equalization algorithm [8].

Algorithm 1: One-scale inverse compositional algorithm

input : $I_1, I_2, \mathbf{p}, \epsilon, j_{max}, \rho$
output: Motion parameter $\mathbf{p} \in \mathbb{R}^n$
 // Precomputations
 1 Estimate the gradient ∇I_1 as in (17) and (18) using central differences with constant boundary condition.
 2 Compute the Jacobian matrix J as in (6) (see Table 1 for typical planar transformations).
 3 Compute $G = \nabla I_1^T J$.
 4 Compute $G^T G$.
 // Incremental refinement
 5 Initialize $\mathbf{p}_0 = \mathbf{p}$ and $j = 1$.
 6 **repeat**
 7 **for** $\mathbf{x} \in \Omega$ **do**
 8 **if** $\Psi(\mathbf{x}; \mathbf{p}_{j-1}) \in [0, M - 1] \times [0, N - 1]$ **then**
 9 | Compute $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1}))$ by bicubic interpolation with constant boundary condition.
 10 **else**
 11 | Set $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1})) = 0$
 12 | Compute $DI(\mathbf{x}) = I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1})) - I_1(\mathbf{x})$.
 13 Update the threshold parameter of the error function ρ as explained in Section 2.3.
 14 Compute $\tilde{\rho}' = \rho'(\|DI\|^2)$.
 15 Compute the vector $\mathbf{b} = \sum_{\mathbf{x} \in \Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T DI(\mathbf{x})$.
 16 Compute the Hessian $H = \sum_{\mathbf{x} \in \Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x})$ and invert it.
 17 Compute $\Delta \mathbf{p}_j = H^{-1} \mathbf{b}$.
 18 $\Psi(\cdot; \mathbf{p}_j) \leftarrow \Psi(\cdot; \mathbf{p}_{j-1}) \circ \Psi(\cdot; \Delta \mathbf{p}_j)^{-1}$.
 19 $j \leftarrow j + 1$.
 20 **until** $j > j_{max}$ **or** $\|\Delta \mathbf{p}_{j-1}\| \leq \epsilon$;
 21 **Return** $\mathbf{p} = \mathbf{p}_j$.

2.3 Error Function

The influence of the error function ρ on the model estimation is only determined by its variations, i.e., its derivative ρ' through the weighting by $\rho'(\|DI(\mathbf{x})\|^2)$ in (13) and (14). In theory, any increasing and derivable function can be chosen as the error function but in the following we only consider the L2 error function, for which the computations are simplified, and the robust error functions, for which the model estimation is robust to outliers. Note that the method can be extended to error functions that are derivable almost everywhere by arbitrarily choosing a representative of the derivative. In particular, it allows to use the truncated L2 error function.

L2 error function. The error function originally considered in [2] was the identity function $\rho(s) = s$, which results in an L2 error in (2). It has the advantage of having a constant derivative $\rho' = 1$ so that the Hessian H , defined in (13), and its inverse can be precomputed before the incremental refinement. Another theoretical advantage is that in Section 2.1 only one first order Taylor expansion is necessary and $E_2 = E_3$. However, as it gives the same weight to every pixel, the model estimation is not robust to outliers.

Robust error function. We call robust error function [1, 17] an error function that reduces the influence of high errors in the model estimation. Typically, it is the case when ρ' is bounded and

$\rho'(s) \xrightarrow{s \rightarrow +\infty} 0$. The model estimation using a robust error function is more robust to outliers because they have less influence. In particular, it allows to deal with problems like occlusions, noise, local brightness changes or spurious motions. Typical robust error functions considered in [17] are given in Table 2. Note that they all depend on a threshold parameter $\lambda > 0$, which controls the variation of the error function, i.e., the influence of outliers. A small value limits the influence of pixels with a high error while a large value means that all the pixels tend to have the same weighting.

Selection of the threshold parameter. In Algorithm 1, when the error function depends on a threshold parameter that is not specified by the user, it is initialized with a large value λ_0 and then geometrically reduced during the incremental refinement. This strategy successively reduces the influence of outliers, which are gradually eliminated. In practice, at step j of the incremental refinement we use the parameter $\lambda_j = \max(0.9^j \lambda_0, 5)$, where $\lambda_0 = 80$.

2.4 Coarse-to-fine Multiscale Approach

The two approximations in Section 2.1 are only valid under the assumption that $\Delta \mathbf{p}$ is small. Therefore, in order to estimate large displacements, a coarse-to-fine multiscale approach is used.

Gaussian pyramid of an image. Let I be an image, N_{scales} be the number of scales in the pyramid and $\eta \in (0, 1)$ be the downsampling factor. For $s \in \{0, \dots, N_{\text{scales}} - 1\}$ we note I^s the image of the pyramid at scale s . The Gaussian pyramid is recursively computed for $s = 1$ to $N_{\text{scales}} - 1$ from $I_0 = I$ by

$$I^s(\mathbf{x}) = (G_\sigma * I^{s-1})\left(\frac{1}{\eta}\mathbf{x}\right). \quad (20)$$

In order to avoid aliasing and to reduce the noise, the images are smoothed with a Gaussian kernel of standard deviation

$$\sigma = \sigma(\eta) = \sigma_0 \sqrt{\frac{1}{\eta^2} - 1}, \quad (21)$$

where $\sigma_0 = 0.6$ is found empirically in [12]. After the convolution (which is computed by applying a discrete kernel with finite support and using the whole-symmetric boundary condition [7]), the images are resampled using bicubic interpolation with a step $\frac{1}{\eta}$. In practice, the number of scales N_{scales} is adjusted so that the coarsest scale image is greater than 32 pixels in the shortest dimension. Thus, the maximal number of scales is

$$N_{\text{scales}}^{\max} = 1 + \left\lceil \frac{\log\left(\frac{\min(M,N)}{32}\right)}{-\log(\eta)} \right\rceil. \quad (22)$$

Coarse-to-fine approach. The estimation of the motion between I_1 and I_2 using the multiscale approach is done as follows. First, the two Gaussian pyramids $(I_1^s)_{0 \leq s \leq N_{\text{scales}} - 1}$ and $(I_2^s)_{0 \leq s \leq N_{\text{scales}} - 1}$ are computed as explained in the previous paragraph. Assuming that the images share a large common part, the motion is initialized at the coarsest scale $s = N_{\text{scales}} - 1$ as $\mathbf{p}^{N_{\text{scales}} - 1} = 0$ (i.e., the identity transformation) and is estimated using the one-scale inverse compositional algorithm (Algorithm 1). Then the estimation is refined in the following finer scales. To transfer the motion parameter \mathbf{p}^s at scale s to the motion parameter \mathbf{p}^{s-1} at scale $s - 1$, the transformation is updated according to the parametrization. Update rules for the parametrizations of planar transformations proposed in Table 1 are presented in Table 3. The multiscale inverse compositional algorithm is shown in Algorithm 2.

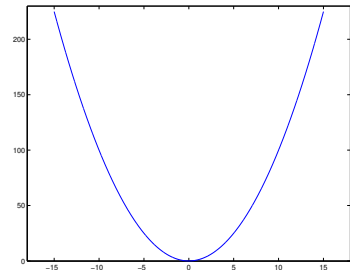
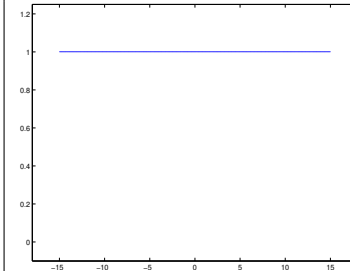
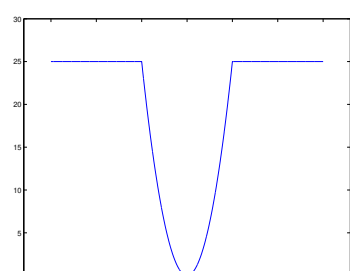
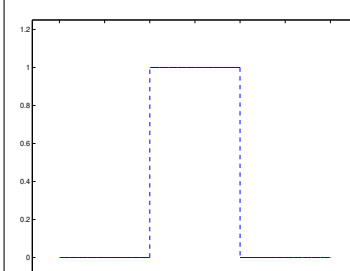
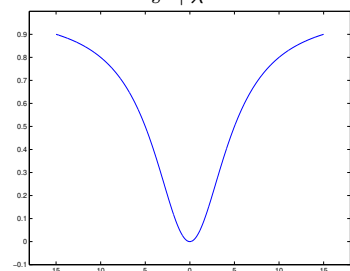
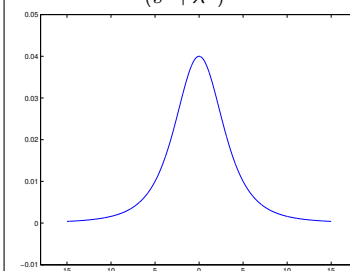
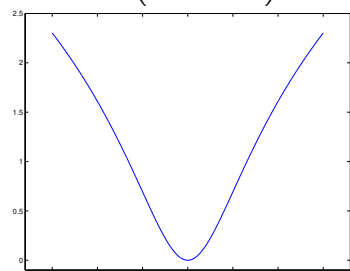
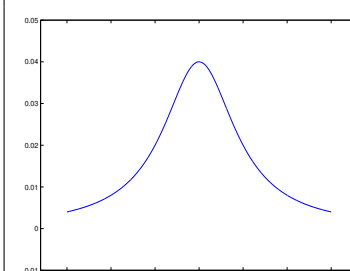
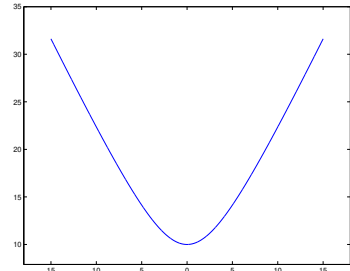
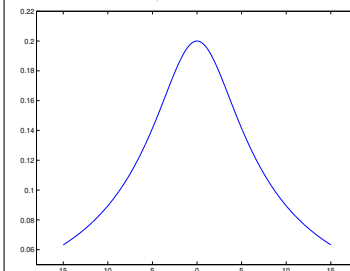
Type	$s \mapsto \rho_\lambda(s^2)$ s^2	$s \mapsto \rho'_\lambda(s^2)$ 1
L2	 A plot of the function $s \mapsto s^2$ for $s \in [-15, 15]$. The curve is a parabola opening upwards with its vertex at (0,0). The y-axis ranges from 0 to 200.	 A plot of the constant function $s \mapsto 1$ for $s \in [-15, 15]$. The curve is a horizontal line at $y=1$. The y-axis ranges from 0 to 1.2.
Truncated L2	$\begin{cases} s^2 & \text{if } s < \lambda \\ \lambda^2 & \text{otherwise} \end{cases}$  A plot of the truncated L2 function for $s \in [-15, 15]$ and $\lambda=5$. The function is s^2 for $ s < 5$ and $\lambda^2=25$ for $ s \geq 5$. The y-axis ranges from 0 to 30.	$\begin{cases} 1 & \text{if } s < \lambda \\ 0 & \text{otherwise} \end{cases}$  A plot of the derivative of the truncated L2 function for $s \in [-15, 15]$ and $\lambda=5$. The function is 1 for $ s < 5$ and 0 for $ s \geq 5$. The y-axis ranges from 0 to 1.2.
Geman & McClure	$\frac{s^2}{s^2 + \lambda^2}$  A plot of the Geman & McClure function for $s \in [-15, 15]$ and $\lambda=5$. The curve is symmetric about $s=0$, with a minimum at (0,0) and asymptotes at $y=1$. The y-axis ranges from -0.1 to 0.9.	$\frac{\lambda^2}{(s^2 + \lambda^2)^2}$  A plot of the derivative of the Geman & McClure function for $s \in [-15, 15]$ and $\lambda=5$. The curve is symmetric about $s=0$, with a peak at (0,0.04) and asymptotes at $y=0$. The y-axis ranges from -0.01 to 0.05.
Lorentzian	$\log\left(1 + \left(\frac{s}{\lambda}\right)^2\right)$  A plot of the Lorentzian function for $s \in [-15, 15]$ and $\lambda=5$. The curve is symmetric about $s=0$, with a minimum at (0,0) and asymptotes at $y=2.5$. The y-axis ranges from 0 to 2.5.	$\frac{1}{s^2 + \lambda^2}$  A plot of the derivative of the Lorentzian function for $s \in [-15, 15]$ and $\lambda=5$. The curve is symmetric about $s=0$, with a peak at (0,0.04) and asymptotes at $y=0$. The y-axis ranges from -0.01 to 0.05.
Charbonnier	$2\sqrt{s^2 + \lambda^2}$  A plot of the Charbonnier function for $s \in [-15, 15]$ and $\lambda=5$. The curve is symmetric about $s=0$, with a minimum at (0,10) and asymptotes at $y=30$. The y-axis ranges from 10 to 30.	$\frac{s}{\sqrt{s^2 + \lambda^2}}$  A plot of the derivative of the Charbonnier function for $s \in [-15, 15]$ and $\lambda=5$. The curve is symmetric about $s=0$, with a peak at (0,0.2) and asymptotes at $y=0$. The y-axis ranges from 0 to 0.22.

Table 2: Example of error functions. Except for the L2 error function, they are all robust error functions depending on a threshold parameter λ . The curves correspond to $\lambda = 5$.

Transform	\mathbf{p}^{s-1}	\mathbf{p}^s
Translation	(t_x, t_y)	$\frac{1}{\eta}(t_x, t_y)$
Euclidean	(t_x, t_y, θ)	$(\frac{1}{\eta}t_x, \frac{1}{\eta}t_y, \theta)$
Similarity	(t_x, t_y, a, b)	$(\frac{1}{\eta}t_x, \frac{1}{\eta}t_y, a, b)$
Affinity	$(t_x, t_y, a_{11}, a_{12}, a_{21}, a_{22})$	$(\frac{1}{\eta}t_x, \frac{1}{\eta}t_y, a_{11}, a_{12}, a_{21}, a_{22})$
Homography	$(h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32})$	$(h_{11}, h_{12}, \frac{1}{\eta}h_{13}, h_{21}, h_{22}, \frac{1}{\eta}h_{23}, \eta h_{31}, \eta h_{32})$

Table 3: Update rule in the coarse-to-fine scheme for the parametrizations of planar transformations proposed in Table 1.

Algorithm 2: Multiscale inverse compositional algorithm

input : $I_1, I_2, \epsilon, j_{\max}, \rho, N_{\text{scales}}, \eta$
output: Transformation $\mathbf{p} \in \mathbb{R}^n$

- 1 Create a Gaussian pyramid of images I_1^s, I_2^s for $s = 0, \dots, N_{\text{scales}} - 1$
- 2 Initialize with $\mathbf{p}^{N_{\text{scales}}-1} = \mathbf{0}$
- 3 **for** $s = N_{\text{scales}} - 1$ **to** 0 **do**
- 4 Compute \mathbf{p}^s with Algorithm 1 applied to $I_1^s, I_2^s, \mathbf{p}^s, \epsilon, j_{\max}, \rho$.
- 5 **if** $s > 0$ **then**
- 6 Compute \mathbf{p}^{s-1} from \mathbf{p}^s by zoom of factor η (see Table 3 for typical planar transformations).
- 7 **Return** $\mathbf{p} = \mathbf{p}^0$

3 Modifications of the Inverse Compositional Algorithm

In this section we propose simple modifications to the inverse compositional algorithm that allow to improve its performance and precision. These improvements are experimentally verified in Section 4.

3.1 Grayscale Conversion

Assume that I_1 and I_2 are color images (i.e. $C = 3$). We consider a classical alternative for color handling, which consists in averaging the channels to obtain a grayscale image. This is valid since the motion is the same for all the channels. This divides by 3 the number of input pixels and by $\sqrt{3} \simeq 1.7$ the noise level. As we will see in Section 4.2.2, there is no clear advantage of using color over grayscale images. Since operating on grayscale images implies less computations, we use it.

3.2 Boundary Handling by Discarding Boundary Pixels

Even though it concerns a relatively small amount of pixels, the handling of boundary pixels has a significant impact on the performance of the algorithm. To estimate $\nabla I_1(\mathbf{x})$ at a pixel \mathbf{x} close to the boundary of Ω , an arbitrary extension of the domain is needed (constant extension in Algorithm 1 and whole-symmetric extension in Section 3.3). Also the evaluation of $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1}))$ by bicubic interpolation requires an arbitrary extension for position $\Psi(\mathbf{x}; \mathbf{p}_{j-1})$ that fall close to the boundary of $[0, M - 1] \times [0, N - 1]$. Therefore, during the motion estimation the boundary pixels are more likely to have incorrect gradient estimates, which deteriorate the performance of the algorithm. When a robust error function is used, the influence of these boundary effects is lessened but is still noticeable.

The handling proposed in [17], and used in Algorithm 1, only avoids the interpolation of values outside of the image domain. Let $\mathbf{x} \in \Omega$ such that $\Psi(\mathbf{x}; \mathbf{p}_{j-1})$ falls outside of the domain $[0, M - 1] \times [0, N - 1]$. Then, the value $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1}))$ is set to 0. This strategy is only adapted to synthetic

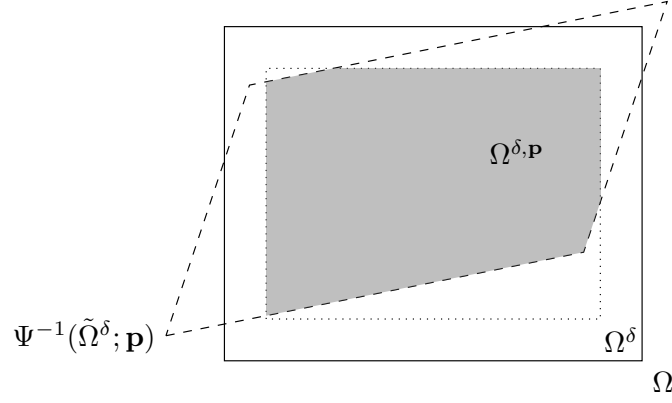


Figure 2: Example of domain $\Omega^{\delta, \mathbf{p}}$. Ω is the spatial domain where I_1 and I_2 are defined. Ω^δ represents the pixels at distance at least δ of the boundary of Ω . Pixels outside of Ω^δ are discarded to avoid incorrect gradient estimations. $\tilde{\Omega}^\delta$ is the continuous domain corresponding to the convex hull of Ω^δ . The continuous domain $\Psi^{-1}(\tilde{\Omega}^\delta; \mathbf{p})$ represents the points whose images by $\Psi(\cdot; \mathbf{p})$ belong to Ω^δ . Pixels outside of $\Omega \cap \Psi^{-1}(\tilde{\Omega}^\delta; \mathbf{p})$ are discarded to avoid incorrect interpolations. Finally, pixels outside of the gray area are discarded and the computations are made on the pixels of $\Omega^{\delta, \mathbf{p}} = \Omega^\delta \cap \Psi^{-1}(\tilde{\Omega}^\delta; \mathbf{p})$.

images where pixels outside the domain are set to 0 during the resampling (see example in Figure 1). Otherwise, as in general $I_1(\mathbf{x})$ has no reason to be close to 0, the model error at \mathbf{x} is likely to be high.

Discarding boundary pixels. We propose an alternative strategy that handles all boundary pixels without introducing outliers. It consists in discarding boundary pixels from the sums in the energies introduced in Section 2.1. Note that it has the advantage of reducing the complexity of the algorithm.

More precisely, let δ be a non-negative integer and define for $\mathbf{p} \in \mathbb{R}^n$

$$\Omega^\delta = \Omega_{M,N}^\delta = \{\delta, \dots, M-1-\delta\} \times \{\delta, \dots, N-1-\delta\}, \quad (23)$$

$$\tilde{\Omega}^\delta = \tilde{\Omega}_{M,N}^\delta = [\delta, M-1-\delta] \times [\delta, N-1-\delta], \quad (24)$$

$$\Omega^{\delta, \mathbf{p}} = \{\mathbf{x} \in \Omega^\delta \mid \Psi(\mathbf{x}; \mathbf{p}) \in \tilde{\Omega}^\delta\} = \Omega^\delta \cap \Psi^{-1}(\tilde{\Omega}^\delta; \mathbf{p}). \quad (25)$$

Ω^δ represents the pixels at distance at least δ of the boundary of Ω . Pixels outside of Ω^δ are discarded to avoid incorrect gradient estimations. $\tilde{\Omega}^\delta$ is the continuous domain corresponding to the convex hull of Ω^δ . The continuous domain $\Psi^{-1}(\tilde{\Omega}^\delta; \mathbf{p})$ represents the points whose images by $\Psi(\cdot; \mathbf{p})$ belong to $\tilde{\Omega}^\delta$. Pixels outside of $\Omega \cap \Psi^{-1}(\tilde{\Omega}^\delta; \mathbf{p}_j)$ are discarded to avoid incorrect interpolations. Finally, the computations are made on the pixels of $\Omega^{\delta, \mathbf{p}} = \Omega^\delta \cap \Psi^{-1}(\tilde{\Omega}^\delta; \mathbf{p})$. We display in Figure 2 an example of domain $\Omega^{\delta, \mathbf{p}}$.

At step j of the incremental refinement, the boundary pixels are assumed to be located in $\Omega \setminus \Omega^{\delta, \mathbf{p}_{j-1}}$. Boundary pixels are discarded by replacing Ω by $\Omega^{\delta, \mathbf{p}_{j-1}}$ in the sums of the energies of Section 2.1. Equivalently it comes back to applying the mask $1_{\Omega^{\delta, \mathbf{p}_{j-1}}}(\mathbf{x})$. Consequently, the increment $\Delta \mathbf{p}_j$ is given by

$$\Delta \mathbf{p}_j = H_\delta^{-1} \mathbf{b}_\delta, \quad (26)$$

where

$$H_\delta = H_\delta(\mathbf{p}_{j-1}) = \sum_{\mathbf{x} \in \Omega^{\delta, \mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x}) = \sum_{\mathbf{x} \in \Omega} 1_{\Omega^{\delta, \mathbf{p}_{j-1}}}(\mathbf{x}) \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x}), \quad (27)$$

$$\mathbf{b}_\delta = \mathbf{b}_\delta(\mathbf{p}_{j-1}) = \sum_{\mathbf{x} \in \Omega^{\delta, \mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T D I(\mathbf{x}) = \sum_{\mathbf{x} \in \Omega} 1_{\Omega^{\delta, \mathbf{p}_{j-1}}}(\mathbf{x}) \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T D I(\mathbf{x}). \quad (28)$$

3.3 Gradient Estimation on a Prefiltered Image

In Algorithm 1, the gradient ∇I_1 is estimated using the central differences scheme defined by (17) and (18). In [15], it was shown that the shift estimation with inverse compositional based-methods can be improved by using other gradient estimators. In particular, the shift estimation becomes more robust under noise. We extend this study to the general context of parametric motion estimation.

Smoothing the input image I_1 reduces its noise and its aliasing, which may lead to a better gradient estimation. Therefore, during the incremental refinement, we do not estimate directly the gradient of I_1 but we replace I_1 and I_2 by prefiltered versions \tilde{I}_1 and \tilde{I}_2 and estimate the gradient $\nabla \tilde{I}_1$. In order to be compatible with the gradient, the difference image DI is replaced by a prefiltered difference image $\tilde{D}I$. Both the gradient estimation and the prefiltering are computed by applying separable kernels. More precisely, a gradient estimation method is determined by a pair of matched prefilter and derivative kernels (stored as vectors):

- the prefilter kernel \mathbf{k} is symmetrical,
- the derivative kernel \mathbf{d} , which is anti-symmetrical.

The prefilter is defined as $\mathbf{k}^T * \mathbf{k}$ while the horizontal and vertical gradient filters are defined respectively by $\mathbf{d}^T * \mathbf{k}$ and $\mathbf{k}^T * \mathbf{d}$. The prefiltered image \tilde{I}_1 whose gradient is estimated is given by

$$\tilde{I}_1 = \mathbf{k}^T * \mathbf{k} * I_1. \quad (29)$$

The gradient $\nabla \tilde{I}_1$ is estimated by computing the partial derivative estimates

$$\frac{\partial \tilde{I}_1}{\partial x} \simeq \mathbf{d}^T * \mathbf{k} * I_1, \quad (30)$$

$$\frac{\partial \tilde{I}_1}{\partial y} \simeq \mathbf{k}^T * \mathbf{d} * I_1. \quad (31)$$

Note that computing the gradient estimation with (30) and (31) does not require \tilde{I}_1 . However it is still computed along with \tilde{I}_2 in order to get the difference image during the incremental refinement. At step j , the prefiltered difference image $\tilde{D}I$ is given by²

$$\tilde{D}I(\mathbf{x}) = \tilde{I}_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1})) - \tilde{I}_1(\mathbf{x}), \quad \mathbf{x} \in \Omega. \quad (32)$$

The convolutions with one-dimensional kernels are computed using the whole-symmetric boundary condition [7]. Let I be an image and \mathbf{k}_1 and \mathbf{k}_2 be two vectors. The filtering $\mathbf{k}_2^T * \mathbf{k}_1 * I$ is computed by convolving each column of I with \mathbf{k}_1 and then convolving each row of the result with \mathbf{k}_2 .

All the considered gradient estimation kernels are shown in Table 4. In practice the kernels \mathbf{k} and \mathbf{d} were designed simultaneously in order to verify given properties [20, 5]. Note that the central differences estimator corresponds to $\mathbf{k} = 1$ and $\mathbf{d} = \frac{1}{2}(-1, 0, 1)^T$ so that there is no prefiltering required. In the following, when the central difference estimator is chosen, the gradient is computed as in Algorithm 1, i.e., with no prefiltering and the constant boundary condition.

3.4 First Scale of the Gaussian Pyramid

When dealing with low quality input images (i.e. noisy), skipping the finest scales in the Gaussian pyramid usually yields results similar to using all the scales in the multiscale algorithm 2. Indeed the images are smoothed during the construction of the Gaussian pyramid reducing noise, alias,

²Interpolating the prefiltered image \tilde{I}_2 should be easier since the image is smooth.

Gradient estimator		Sample number				
		-2	-1	0	1	2
Central differences	k			1		
	d		-0.5	0	0.5	
Hypomode	k			0.5	0.5	
	d			-1	1	
Farid 3×3	k		0.229879	0.540242	0.229879	
	d		-0.425287	0	0.425287	
Farid 5×5	k	0.037659	0.249153	0.426375	0.249153	0.037659
	d	-0.109604	-0.276691	0	0.276691	0.109604
Gaussian $\sigma = 0.3$	k		0.003865	0.999990	0.003865	
	d		-0.707110	0	0.707110	
Gaussian $\sigma = 0.6$	k	0.003645	0.235160	0.943070	0.235160	0.003645
	d	-0.021915	-0.706770	0	0.706770	0.021915

 Table 4: Prefilter kernel **k** and derivative kernel **d** of the proposed gradient estimators.

chromatic aberration, and zipper effects [10]. The motion estimation at coarser scales is less affected by the artifacts of the input images and the improvement at the finer scales can be negligible. Another advantage of not using the finest scales is that it significantly reduces the complexity of the algorithm. Because of its recursive construction the whole Gaussian pyramid has to be computed, but the motion estimation is only performed at the coarser scales.

In order to select the first scale used in the Gaussian Pyramid, we introduce a new parameter $s_0 \in \{0, \dots, N_{\text{scales}} - 1\}$. The modified multiscale approach only refines the motion estimation for the scales $N_{\text{scales}} - 1$ to s_0 . Note that if $s_0 \geq 2$, \mathbf{p}^0 is computed from \mathbf{p}^{s_0-1} thanks to the update rule with zoom factor η^{s_0-1} (see Table 3).

3.5 Modified Inverse Compositional Algorithm

Incorporating the proposed modifications to Algorithms 1 and 2, we obtain the modified inverse compositional algorithm. The one-scale and multiscale versions are respectively presented in Algorithms 3 and 4. The additional parameters are:

1. the non-negative integer δ for discarding boundary pixels (see Section 3.2),
2. the pair (\mathbf{k}, \mathbf{d}) of kernels for the gradient estimation and the prefiltering (see Section 3.3),
3. the first scale $s_0 \in \{0, \dots, N_{\text{scales}} - 1\}$ used in the pyramid (see Section 3.4).

In addition, the user has to specify if the grayscale conversion of Section 3.1 is used.

4 Experiments

In this section, we evaluate experimentally the impact of the modifications proposed in Section 3 on the motion estimation performance and on the computation time. First, we describe the experimental setup and the error measure used to evaluate the performance on synthetic data. Then we study the influence of the modifications and show to what extent each of them improves the performance of the algorithm. Finally, we compare the non-modified and modified inverse compositional algorithms with a classic parametric motion estimation based on the SIFT keypoints and the RANSAC algorithm.

Algorithm 3: One-scale modified inverse compositional algorithm

input : $I_1, I_2, \mathbf{p}, \epsilon, j_{\max}, \rho, \delta, (\mathbf{k}, \mathbf{d})$
output: Motion parameter $\mathbf{p} \in \mathbb{R}^n$
// Precomputations

- 1 Estimate the gradient $\nabla \tilde{I}_1$ from I_1 and (\mathbf{k}, \mathbf{d}) as in (30) and (31).
- 2 Compute the prefiltered images $\tilde{I}_1 = \mathbf{k}^T * \mathbf{k} * I_1$ and $\tilde{I}_2 = \mathbf{k}^T * \mathbf{k} * I_2$.
- 3 Compute the Jacobian matrix J as in (6) (see Table 1 for typical planar transformations).
- 4 Compute $G = \nabla \tilde{I}_1^T J$.
- 5 Compute $G^T G$.

// Incremental refinement

- 6 Initialize $\mathbf{p}_0 = \mathbf{p}$ and $j = 1$.
- 7 **repeat**
- 8 Compute the domain $\Omega^{\delta, \mathbf{p}_{j-1}}$ as in (25).
- 9 **for** $\mathbf{x} \in \Omega^{\delta, \mathbf{p}_{j-1}}$ **do**
- 10 Compute $\tilde{I}_2(\Psi(\mathbf{x}, \mathbf{p}_{j-1}))$ by bicubic interpolation with constant boundary condition.
- 11 Compute $\tilde{D}I(\mathbf{x}) = \tilde{I}_2(\Psi(\mathbf{x}, \mathbf{p}_{j-1})) - \tilde{I}_1(\mathbf{x})$.
- 12 Update the threshold parameter of the error function ρ as explained in Section 2.3.
- 13 Compute $\tilde{\rho}' = \rho'(\|\tilde{D}I\|^2)$ on $\Omega^{\delta, \mathbf{p}_{j-1}}$.
- 14 Compute the vector $\mathbf{b}_\delta = \sum_{\mathbf{x} \in \Omega^{\delta, \mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T \tilde{D}I(\mathbf{x})$.
- 15 Compute the Hessian $H_\delta = \sum_{\mathbf{x} \in \Omega^{\delta, \mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x})$ and invert it.
- 16 Compute $\Delta \mathbf{p}_j = H_\delta^{-1} \mathbf{b}_\delta$.
- 17 $\Psi(\cdot; \mathbf{p}_j) \leftarrow \Psi(\cdot; \mathbf{p}_{j-1}) \circ \Psi(\cdot; \Delta \mathbf{p}_j)^{-1}$.
- 18 $j \leftarrow j + 1$.
- 19 **until** $j > j_{\max}$ **or** $\|\Delta \mathbf{p}_{j-1}\| \leq \epsilon$;
- 20 Return $\mathbf{p} = \mathbf{p}_j$.

Algorithm 4: Modified multiscale inverse compositional algorithm

input : $I_1, I_2, \epsilon, j_{\max}, \rho, N_{\text{scales}}, s_0 \in \{0, \dots, N_{\text{scales}} - 1\}, \eta, \delta, (\mathbf{k}, \mathbf{d})$
output: Transformation $\mathbf{p} \in \mathbb{R}^n$

- 1 If the grayscale conversion is specified by the user, replace I_1 and I_2 by the average of their channels.
- 2 Create a Gaussian pyramid of images I_1^s, I_2^s for $s = 0, \dots, N_{\text{scales}} - 1$
- 3 Initialize with $\mathbf{p}^{N_{\text{scales}}-1} = 0$
- 4 **for** $s = N_{\text{scales}} - 1$ **to** s_0 **do**
- 5 Compute \mathbf{p}^s with Algorithm 3 applied to $I_1^s, I_2^s, \mathbf{p}^s, \epsilon, j_{\max}, \rho, \delta, (\mathbf{k}, \mathbf{d})$.
- 6 **if** $s > 0$ **then**
- 7 Compute \mathbf{p}^{s-1} from \mathbf{p}^s by zoom of factor η (see Table 3 for typical planar transformations).
- 8 **if** $s_0 > 1$ **then**
- 9 Compute \mathbf{p}^0 from \mathbf{p}^{s_0-1} by zoom of factor η^{s_0-1} (see Table 3 for typical planar transformations).
- 10 Return $\mathbf{p} = \mathbf{p}^0$

4.1 Experimental Setup

To evaluate the performance of a motion estimation method we build, from a reference image, a sequence of noisy warped images whose transformations are known. The role of the reference and warped images is swapped in the estimation algorithm to avoid the accumulation of interpolation error. For each image, the error between the estimated transformation and the ground truth transformation is expressed in terms of end-point error. The mean of the error among the sequence gives an evaluation of the performance of the method.

End-point error. Let I_1 and I_2 be two images linked by a transformation parametrized by \mathbf{p}^* . Let \mathbf{p} be the estimated motion parameters provided by a given motion estimation method. The end-point error $\text{EPE}(\mathbf{p}, \mathbf{p}^*)(\mathbf{x})$ at a pixel $\mathbf{x} \in \Omega$ is defined by

$$\text{EPE}(\mathbf{p}, \mathbf{p}^*)(\mathbf{x}) = \|\Psi(\mathbf{x}; \mathbf{p}) - \Psi(\mathbf{x}; \mathbf{p}^*)\|_2. \quad (33)$$

It is the distance between the images of the estimated transformation and the ground truth transformation. The end-point error is a measure of the error that is commonly used in optical flow estimation [18, 19]. An example of end-point error field, i.e the image of end-point errors on Ω , is shown in Figure 3. The average end-point error $\overline{\text{EPE}}(\mathbf{p}, \mathbf{p}^*)$ is defined as the mean of the end-point errors $\text{EPE}(\mathbf{p}, \mathbf{p}^*)(\mathbf{x})$ over the domain Ω , i.e.

$$\overline{\text{EPE}}(\mathbf{p}, \mathbf{p}^*) = \frac{1}{MN} \sum_{\mathbf{x} \in \Omega} \text{EPE}(\mathbf{p}, \mathbf{p}^*)(\mathbf{x}). \quad (34)$$

Building the test sequence. As the error varies with the images, the transformations and the noise, we evaluate the performance of methods by considering the mean of the errors over a sequence of images, which is built as follows. Let I be a reference input image. We draw N_{images} homographies parametrized by $(\mathbf{p}_i^*)_{1 \leq i \leq N_{\text{images}}}$ by randomly shifting the four corners of the domain Ω along both directions. The shifts are drawn independently and uniformly in $[-L, L]$ for a given non-negative integer L . We build a sequence $(I^i)_{1 \leq i \leq N_{\text{images}}}$ of warped images using bicubic interpolation (with whole-symmetric boundary condition), verifying $I^i = I(\Psi(\cdot; \mathbf{p}_i^*))$. Finally, we add Gaussian white noise of standard deviation σ to the reference image and the warped images. The role of the reference and warped images is swapped in the estimation algorithm to avoid the accumulation of interpolation error. In other words, we use $I_1 = I^i$ and $I_2 = I$.

We compute the average error $\overline{\text{EPE}}_i$ as in (34). The error of the method for the image I and the noise level σ , noted EPE to simplify, is evaluated as the mean

$$\text{EPE} = \frac{1}{N_{\text{images}}} \sum_{i=1}^{N_{\text{images}}} \overline{\text{EPE}}_i. \quad (35)$$

Note that this error is not deterministic since it depends on the transformations and the noise realizations.

4.2 Influence of the Modifications

In order to evaluate the influence of the modifications proposed in Section 3, we use the experimental setup described in Section 4.1. We introduce one by one the modifications to simplify the presentation of the results.


 (a) Reference image $I_1 = I_2(\Psi(\cdot; \mathbf{p}^*))$

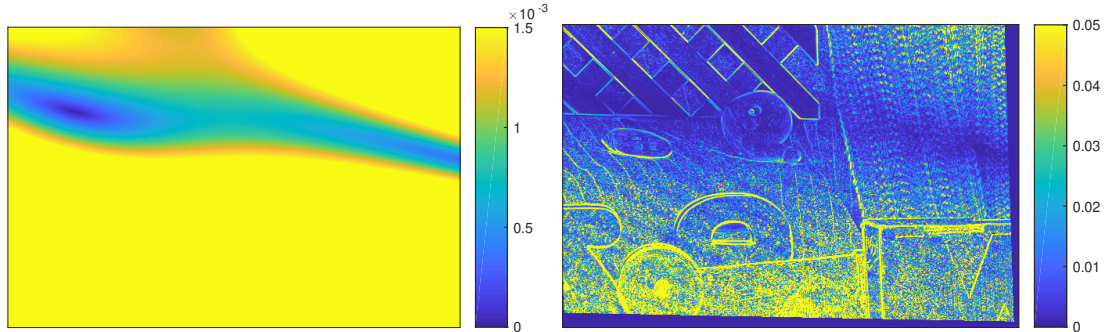
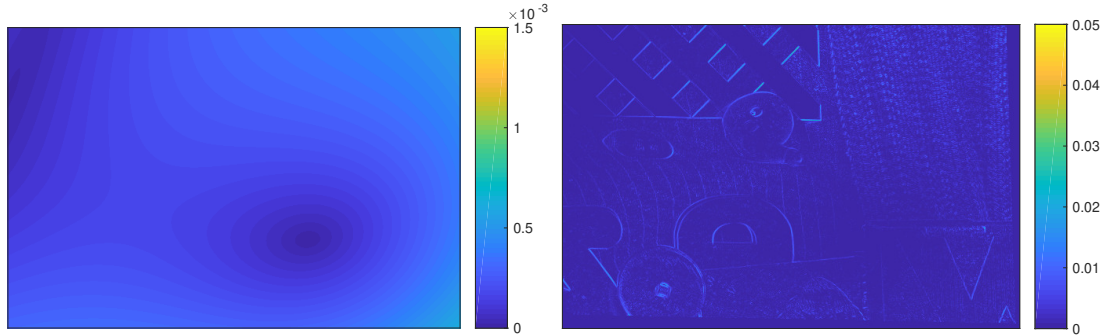
 (b) Warped image I_2

 (c) End-point error field $EPE(\mathbf{p}, \mathbf{p}^*)(\mathbf{x})$ (left) and residual (right) using the IC algorithm

 (d) End-point error field $EPE(\mathbf{p}, \mathbf{p}^*)(\mathbf{x})$ (left) and residual (right) using the modified IC algorithm

Figure 3: Example of motion estimation on synthetic data. The warped image I_2 is the image used in the experiments of Section 4.2. It is a color image of size 584×388 taken from the Middlebury database [4]. The reference image $I_1 = I_2(\Psi(\cdot; \mathbf{p}^*))$ is an example of synthetic image related to I_2 by an homography and obtained by bicubic interpolation with whole-symmetric boundary condition. On the second and third line, the estimated motion \mathbf{p} is obtained using either the inverse compositional algorithm or the modified inverse compositional algorithm. For the modifications we use $\delta = 5$, the Farid 5×5 kernel estimator, the grayscale conversion and $s_0 = 0$. For both methods, the right image is actually the root mean square over the channels of the residual $I_1 - I_2(\Psi(\cdot; \mathbf{p}))$, which is obtained by bicubic interpolation. Without modification, we have $EPE=0.00460$ and $RMSE(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x}; \mathbf{p}))) = 0.042838$. With modification, we have $EPE=0.00022$ and $RMSE(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x}; \mathbf{p}))) = 0.001790$.

We take as reference image I the image presented in Figure 3, which is a color image of size 584×388 taken from the Middlebury database [4]. For all the experiments, the sequence of transformations used is the same and is obtained by taking $N_{\text{images}} = 1000$ and $L = 20$. The noise level σ varies in $\{0, 3, 5, 10, 20, 30, 50\}$. In order to perform reliable comparisons, for a given noise level, the noisy images used are the same for all the methods. We use the following parameter values: $\eta = \frac{1}{2}$, $\epsilon = 0.001$, $j_{\text{max}} = 30$ and $N_{\text{scales}} = N_{\text{scales}}^{\text{max}}$ (see (22)). The error functions used are the L2 and the Lorentzian functions, for which the threshold parameter varies during the incremental refinement as explained in Section 2.3. In order to study the performance of each method, we consider the end-point error (EPE) and the computation time. The displayed computation time corresponds to the CPU time used for the $N_{\text{images}} = 1000$ motion estimations and is expressed in seconds. Note that it also corresponds to the average computation time per image in milliseconds. The experiments were made using an Intel(R) Xeon(R) CPU E5-2650 on a single thread.

4.2.1 Discarding Boundary Pixels

To analyze the influence of the boundary handling, we compare the results of the inverse compositional algorithm described in Section 2 and of the modified version that discards boundary pixels with $\delta \in \{0, 5\}$. The gradient estimation is done using the central difference scheme, all scales are used and there is no grayscale conversion. Note that $\delta = 0$ corresponds to the case where outside pixels are discarded during the interpolation (instead of setting the interpolated values to 0). Considering that, in the following, we use the gradient kernels of Table 4 and bicubic interpolation, the value $\delta = 5$ is large enough to discard all boundary pixels.

The results are presented in Table 5. It clearly shows that discarding boundary pixels always provides significantly better results in terms of precision and computation time. Discarding boundary pixels with $\delta = 5$ provides slightly better results than with $\delta = 0$ since it handles all the boundary pixels and not only outside pixels during the interpolation (which introduce more error than inside boundary pixels). By discarding boundary pixels, the incremental refinement is not perturbed by arbitrarily introduced outliers so that it has the following consequences.

On the precision. The precision of the estimated model is increased by a factor ranging from 3.7 to 780 for the L2 function and from 1.4 to 22 for the Lorentzian function. Since the robust error functions handle more correctly the outliers, the precision improvements are less important for the Lorentzian error function than for the L2 function. The improvement factor decreases as the noise level increases i.e. as the noise becomes the main source of estimation error. This explains why the ranges of improvement factor are large.

On the computation time. The computation time is divided by a factor ranging from 1.1 to 2 for the L2 function and from 1.1 to 2.4 for the Lorentzian function. At each step the incremental refinement complexity is lessened but it is not sufficient to explain such a reduction. The main reason is that less iterations are required to converge since the incremental refinement does not try to fit the model with the outliers. Globally the improvement factor tends to decrease with the noise level.

Finally, because it improves the precision and the computation time, we strongly recommend to discard boundary pixels with $\delta = 5$. It is done in the following experiments.

4.2.2 Color Handling

To analyze the influence of the grayscale conversion, we compare the results of the modified inverse compositional algorithm with and without grayscale conversion. Boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used.

		L2			Lorentzian		
		IC	$\delta = 0$	$\delta = 5$	IC	$\delta = 0$	$\delta = 5$
$\sigma = 0$	EPE	0.06268	0.00008	0.00008	0.00221	0.00010	0.00010
	Time	438	320	311	915	419	404
$\sigma = 3$	EPE	0.06296	0.00208	0.00192	0.00341	0.00207	0.00192
	Time	453	333	323	893	387	378
$\sigma = 5$	EPE	0.06394	0.00338	0.00282	0.00517	0.00337	0.00282
	Time	524	373	355	903	402	391
$\sigma = 10$	EPE	0.06646	0.00962	0.00711	0.01163	0.00963	0.00714
	Time	669	406	386	951	481	459
$\sigma = 20$	EPE	0.09194	0.02901	0.01730	0.03340	0.03101	0.02147
	Time	1104	640	544	1290	1109	1069
$\sigma = 30$	EPE	0.14683	0.06508	0.03393	0.06763	0.06569	0.04447
	Time	1319	1008	827	1741	1651	1581
$\sigma = 50$	EPE	0.24210	0.12385	0.06515	0.10959	0.10487	0.07646
	Time	1507	1454	1313	2139	2000	1918

Table 5: Influence of the boundary handling. Comparison between the inverse compositional algorithm presented in Section 2 (noted IC) and the modified inverse compositional algorithm that discards boundary pixels with $\delta = 0$ and $\delta = 5$. The gradient estimation is done using the central difference scheme, all scales are used and there is no grayscale conversion. It clearly shows that discarding boundary pixels always provides significantly better results in terms of precision and computation time. The gain is less and less important as the noise level increases.

The results are presented in Table 6. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.4 to 2.6. Indeed, a reduction was expected since the input number of channels is divided by 3. For large noise value the reduction is globally more important because the input noise is divided by $\sqrt{3}$. For the precision, the results are similar but the grayscale conversion provides the best results for large noise values.

Note that the precision is similar as long as the main structures of the image are preserved by the grayscale conversion. This behavior is confirmed by the experiments presented in Appendix A where additional images were used. On the other hand, we have introduced independent Gaussian noise in each channel, which may not be realistic. For a real image the noise reduction due to the grayscale conversion may not be so important. Nevertheless, we recommend the use of the grayscale conversion because it is much faster and the difference in precision is usually not remarkable.

4.2.3 Gradient Estimation on a Prefiltered Image

To analyze the influence of the gradient estimation, we compare the results of the modified inverse compositional algorithm using each one of the gradient kernels of Table 4. Boundary pixels are discarded with $\delta = 5$, all scales and the grayscale conversion are used.

The results are presented in Table 7 for the L2 error function and in Table 8 for the Lorentzian error function. The results analysis is not as simple as for the boundary handling influence and the color handling. In general, all the estimators provide similar results in terms of precision (except for the hypomode estimator that gives worse results). However, the central differences estimator provides slightly better results for small noise level, while the Farid 3×3 and 5×5 estimators [5] are better for larger noise levels. The main difference between the gradient estimators lies in the computation time. For low noise levels the computation times are similar but for large noise levels the Farid 5×5 estimator provides significantly better results.

In general, computing the gradient on a prefiltered image provides a gradient estimation more

		L2		Lorentzian	
Color handling		Color	Grayscale	Color	Grayscale
$\sigma = 0$	EPE	0.00008	0.00008	0.00010	0.00010
	Time	311	176	404	261
$\sigma = 3$	EPE	0.00192	0.00216	0.00192	0.00215
	Time	323	208	378	262
$\sigma = 5$	EPE	0.00282	0.00300	0.00282	0.00300
	Time	355	195	391	261
$\sigma = 10$	EPE	0.00711	0.00707	0.00714	0.00707
	Time	386	210	459	281
$\sigma = 20$	EPE	0.01730	0.01810	0.02147	0.01929
	Time	544	255	1069	449
$\sigma = 30$	EPE	0.03393	0.03299	0.04447	0.03941
	Time	827	319	1581	806
$\sigma = 50$	EPE	0.06515	0.06192	0.07646	0.07060
	Time	1313	527	1918	1066

Table 6: Influence of the color handling. Boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.4 to 2.6. Indeed, a reduction was expected since the input number of channels is divided by 3. For large noise value the reduction is globally more important because the input noise is divided by $\sqrt{3}$. For the precision, the results are similar but the grayscale conversion provides the best results for large noise values.

robust to noise. In addition, the prefiltered images contain less aliasing and the interpolated values are computed more precisely. Finally, it allows for a faster convergence of the incremental refinement and a more precise motion estimation. Therefore, in the following we use the Farid 5×5 estimator.

		Central Differences	Hypomode	Farid 3×3	Farid 5×5	Gaussian 3	Gaussian 6
$\sigma = 0$	EPE	0.00008	0.03146	0.00017	0.00026	0.00019	0.00015
	Time	176	210	187	202	213	207
$\sigma = 3$	EPE	0.00216	0.03169	0.00241	0.00269	0.00219	0.00235
	Time	208	249	216	206	216	208
$\sigma = 5$	EPE	0.00300	0.03212	0.00317	0.00351	0.00305	0.00312
	Time	195	230	202	205	221	209
$\sigma = 10$	EPE	0.00707	0.03451	0.00702	0.00749	0.00707	0.00694
	Time	210	252	220	210	240	226
$\sigma = 20$	EPE	0.01810	0.04133	0.01698	0.01782	0.01800	0.01694
	Time	255	306	245	238	281	252
$\sigma = 30$	EPE	0.03299	0.04992	0.02917	0.02941	0.03243	0.02940
	Time	319	375	291	258	349	290
$\sigma = 50$	EPE	0.06192	0.06848	0.04644	0.04491	0.05804	0.04788
	Time	527	568	435	370	572	450

Table 7: Influence of the gradient estimator (L2 error function). Comparison of the modified inverse compositional algorithm using the L2 error function and each one of the gradient kernels of Table 4. Boundary pixels are discarded with $\delta = 5$, all scales and the grayscale conversion are used. In general, all the estimators provide similar results in terms of precision (except for the hypomode estimator that gives worse results). However, the central differences estimator provides slightly better results for small noise level, while the Farid 3×3 and 5×5 estimators are better for larger noise levels. For low noise levels the computation times are similar but for large noise levels the Farid 5×5 estimator provides significantly better results.

		Central Differences	Hypomode	Farid 3×3	Farid 5×5	Gaussian 3	Gaussian 6
$\sigma = 0$	EPE	0.00010	0.03140	0.00016	0.00024	0.00010	0.00012
	Time	261	277	251	252	279	261
$\sigma = 3$	EPE	0.00215	0.03162	0.00240	0.00268	0.00217	0.00234
	Time	262	285	255	254	284	255
$\sigma = 5$	EPE	0.00300	0.03203	0.00316	0.00349	0.00303	0.00311
	Time	261	297	254	253	283	262
$\sigma = 10$	EPE	0.00707	0.03437	0.00700	0.00746	0.00705	0.00692
	Time	281	341	272	262	318	285
$\sigma = 20$	EPE	0.01929	0.04145	0.01693	0.01778	0.01906	0.01698
	Time	449	452	334	303	514	361
$\sigma = 30$	EPE	0.03941	0.05042	0.02934	0.02933	0.03638	0.03091
	Time	806	739	481	375	814	617
$\sigma = 50$	EPE	0.07060	0.07333	0.04934	0.04717	0.06203	0.05439
	Time	1066	940	837	704	1041	942

Table 8: Influence of the gradient estimator (Lorentzian error function). Comparison of the modified inverse compositional algorithm using the Lorentzian error function and each one of the gradient kernels of Table 4. Boundary pixels are discarded with $\delta = 5$, all scales and the grayscale conversion are used. In general, all the estimators provide similar results in terms of precision (except for the hypomode estimator that gives worse results). However, the central differences estimator provides slightly better results for small noise level, while the Farid 3×3 and 5×5 estimators are better for larger noise levels. For low noise levels the computation times are similar but for large noise levels the Farid 5×5 estimator provides significantly better results.

4.2.4 First Scale in the Gaussian Pyramid

To analyze the influence of the first scale s_0 used in the Gaussian pyramid, we compare the results of the modified inverse compositional algorithm using $s_0 \in \{0, 1, 2, 3\}$. Boundary pixels are discarded with $\delta = 5$, the Farid 5×5 gradient estimator and the grayscale conversion are used. The results are presented in Table 9 for the L2 error function and in Table 10 for the Lorentzian error function.

Evolution with the number of skipped scales s_0 . As expected, the precision and the computation time both decrease with s_0 , i.e. the number of scales skipped. There is a trade-off between precision and speed. By comparing the evolution between the column s_0 and $s_0 + 1$ we notice that it is less and less interesting to remove scales. Indeed, the decreasing factor for the computation time decreases with s_0 while the increasing factor for the estimation error increases with s_0 . Therefore it is reasonable to only consider $s_0 \in \{0, 1\}$.

Evolution with the noise level σ . As σ increases, it is more and more interesting to skip the finest scale, i.e. to take $s_0 = 1$. Indeed, the decreasing factor for the computation time increases with σ while the increasing factor for the estimation error decreases with σ . For large values of σ , the estimation error is similar for $s_0 = 0$ and $s_0 = 1$ while the computation time is divided by a factor up to 2.3 for the L2 function and 4.4 for the Lorentzian function.

Finally, we recommend to use $s_0 \in \{0, 1\}$ with a choice depending on the context of application and the aim of the user (precision or speed).

		$s_0 = 0$	$s_0 = 1$	$s_0 = 2$	$s_0 = 3$
$\sigma = 0$	EPE	0.00026	0.00328	0.01265	0.05672
	Time	202	131	107	97
$\sigma = 3$	EPE	0.00269	0.00562	0.01446	0.05829
	Time	206	130	110	104
$\sigma = 5$	EPE	0.00351	0.00684	0.01777	0.06094
	Time	205	128	109	107
$\sigma = 10$	EPE	0.00749	0.01261	0.02611	0.06899
	Time	210	132	113	104
$\sigma = 20$	EPE	0.01782	0.02641	0.04101	0.09040
	Time	238	132	114	107
$\sigma = 30$	EPE	0.02941	0.04276	0.07127	0.12303
	Time	258	131	112	106
$\sigma = 50$	EPE	0.04491	0.06679	0.10165	0.19240
	Time	370	160	129	123

Table 9: Influence of the first scale s_0 used in the Gaussian pyramid (L2 error function). Boundary pixels are discarded with $\delta = 5$, the Farid 5×5 gradient estimator and the grayscale conversion are used. The precision and the computation time both decrease with s_0 . There is a trade-off between precision and speed. By comparing the evolution between the column s_0 and $s_0 + 1$ we notice that it is less and less interesting to remove scales. As the noise level σ increases, it is more and more interesting to skip the finest scale, i.e. to take $s_0 = 1$. For large values of σ , the estimation error is similar for $s_0 = 0$ and $s_0 = 1$ while the computation time is divided by a factor up to 2.3.

		$s_0 = 0$	$s_0 = 1$	$s_0 = 2$	$s_0 = 3$
$\sigma = 0$	EPE	0.00024	0.00327	0.01265	0.05667
	Time	252	122	92	87
$\sigma = 3$	EPE	0.00268	0.00561	0.01446	0.05826
	Time	254	140	114	110
$\sigma = 5$	EPE	0.00349	0.00682	0.01777	0.06101
	Time	253	143	119	107
$\sigma = 10$	EPE	0.00746	0.01257	0.02609	0.06901
	Time	262	142	115	106
$\sigma = 20$	EPE	0.01778	0.02639	0.04099	0.09019
	Time	303	146	112	110
$\sigma = 30$	EPE	0.02933	0.04272	0.07131	0.12286
	Time	375	150	116	109
$\sigma = 50$	EPE	0.04717	0.06675	0.10165	0.19220
	Time	704	158	116	105

Table 10: Influence of the first scale used in the Gaussian pyramid (Lorentzian error function). Boundary pixels are discarded with $\delta = 5$, the Farid 5×5 gradient estimator and the grayscale conversion are used. The precision and the computation time both decrease with s_0 . There is a trade-off between precision and speed. By comparing the evolution between the column s_0 and $s_0 + 1$ we notice that it is less and less interesting to remove scales. As the noise level σ increases, it is more and more interesting to skip the finest scale, i.e. to take $s_0 = 1$. For large values of σ , the estimation error is similar for $s_0 = 0$ and $s_0 = 1$ while the computation time is divided by a factor up to 4.4.

4.2.5 Summary of the Improvements

The study of the modifications proposed previously can be summarized as follows:

- Boundary pixels must be discarded with $\delta = 5$.
- The grayscale conversion must be used for high noise level and should be used in general since the eventual loss in precision is negligible with respect to the gain in speed.
- The central differences and the Farid 5×5 gradient estimators provide similar results. But the Farid 5×5 estimator is preferred because it is more robust to noise.
- Discarding the finest scale, i.e. taking $s_0 = 1$, allows for a significant speed up of the motion estimation with a moderate loss of precision, which decreases with the noise level.

Using these recommended modifications:

- Using $s_0 = 0$. The computation time is at least reduced by a factor 2.2. The estimation accuracy is at least improved by a factor 5 for the L2 function and 1.3 for the Lorentzian function.
- Using $s_1 = 1$. The computation time is at least reduced by a factor 3.4. For the L2 function, the estimation accuracy is at least improved by a factor 3.4. For the Lorentzian function the estimation accuracy is at most reduced by a factor 0.6.

4.3 Comparison with a SIFT+RANSAC Based Algorithm

In order to put in perspective the performance of the proposed modified inverse compositional algorithm, it is compared with a classical feature-based motion estimation algorithm. It uses as features the SIFT keypoints [9, 16] and estimates the model with a RANSAC algorithm [6]. Note that the grayscale conversion is used. A similar algorithm can be found in [14].

For the comparison, we consider the SIFT+RANSAC algorithm, the inverse compositional (IC) algorithm described in Algorithm 2 and the modified inverse compositional (mIC) algorithm. For the modifications we follow the recommendations of Section 4.2.5, i.e., we discard boundary pixels with $\delta = 5$ and use the Farid 5×5 kernel estimator, the grayscale conversion and $s_0 \in \{0, 1\}$.

4.3.1 Synthetic Data

First we compare the algorithms using the same experimental setup and synthetic data as in Section 4.2. The results are presented in Table 11.

On the computation time. The inverse compositional algorithm, modified or not, is faster than the SIFT+RANSAC algorithm. The computation time increases with the noise level while it decreases for the SIFT+RANSAC algorithm. Using $s_0 = 0$, the mIC algorithm is 5.6 to 11 times faster for the L2 function and 3 to 8.6 times faster for the Lorentzian function. Using $s_0 = 1$, the mIC algorithm is 13 to 17 times faster for the L2 function and 13 to 18 times faster for the Lorentzian function.

On the precision. The IC algorithm is in general more accurate than the SIFT+RANSAC algorithm (except when using the L2 function for low noise levels because of the incorrect boundary handling). On the opposite, the mIC algorithm always clearly outperforms the SIFT+RANSAC algorithm and in particular for high noise levels. The precision is 14 to 130 times better for $s_0 = 0$ and 7 to 31 times better for $s_0 = 1$.

General remarks on the experiments. We observed that the behavior of the algorithm for other reference images is similar to the reported results. However the improvement factors may differ with the input images. Also, we noticed that in the context of low quality input images the precision may be better while using $s_0 = 1$.

In our tests the reference and warped images are linked with moderate deformations without occlusion nor contrast change. For larger deformations, the inverse compositional based algorithms may be outperformed by feature-based methods. The occlusions highly increase the error for the L2 error function but are well handled by the robust error functions.

		L2				Lorentzian		
		SIFT+R	IC	mIC $s_0 = 0$	mIC $s_0 = 1$	IC	mIC $s_0 = 0$	mIC $s_0 = 1$
$\sigma = 0$	EPE	0.03131	0.06268	0.00026	0.00328	0.00221	0.00024	0.00327
	Time	2171	438	202	131	915	252	122
$\sigma = 3$	EPE	0.04063	0.06296	0.00269	0.00562	0.00341	0.00268	0.00561
	Time	2173	453	206	130	893	254	140
$\sigma = 5$	EPE	0.04805	0.06394	0.00351	0.00684	0.00517	0.00349	0.00682
	Time	2144	524	205	128	903	253	143
$\sigma = 10$	EPE	0.15998	0.06646	0.00749	0.01261	0.01163	0.00746	0.01257
	Time	2208	669	210	132	951	262	142
$\sigma = 20$	EPE	0.82330	0.09194	0.01782	0.02641	0.03340	0.01778	0.02639
	Time	2198	1104	238	132	1290	303	146
$\sigma = 30$	EPE	0.67803	0.14683	0.02941	0.04276	0.06763	0.02933	0.04272
	Time	2177	1319	258	131	1741	375	150
$\sigma = 50$	EPE	1.22713	0.24210	0.04491	0.06679	0.10959	0.04717	0.06675
	Time	2075	1507	370	160	2139	704	158

Table 11: Comparison of motion estimation methods: SIFT+RANSAC (SIFT+R), inverse compositional (IC) algorithm described in Algorithm 2 and the modified inverse compositional (mIC) algorithm with $s_0 \in \{0, 1\}$. The mIC algorithm discards boundary pixels with $\delta = 5$ and uses the Farid 5×5 kernel estimator and the grayscale conversion. The inverse compositional algorithm, modified or not, is always faster than the SIFT+RANSAC algorithm. The computation time increases with the noise level while it decreases for the SIFT+RANSAC algorithm. The IC algorithm is in general more accurate than the SIFT+RANSAC algorithm (except using the L2 function for low noise levels because of the incorrect boundary handling). On the opposite, the mIC algorithm always clearly outperforms the SIFT+RANSAC algorithm and in particular for high noise levels.

4.3.2 Real Data

Secondly, we compare the methods on two images taken from the "Lunch Room" sequence of the PASSTA Dataset [13]. Between the acquisitions, the camera was rotated around its optical center so that the images are linked by an homography. The images I_1 and I_2 , of size 2048×2048 and stored in the JPEG format, are displayed in Figure 4. For the IC and mIC algorithms, we only use the Lorentzian error function.

Since the real motion \mathbf{p}^* is unknown, the end-point error cannot be computed. Let denote by \mathbf{p} the estimated motion by a given method. The algorithm precision is indirectly evaluated by considering the residual $I_1 - I_2(\Psi(\cdot; \mathbf{p}))$, which is computed by bicubic interpolation. The root mean square over the channels of the residuals are displayed in Figure 5. The residuals are inevitably corrupted by JPEG artifacts, noise and interpolation error. The JPEG artifacts and noise are noticeable on flat areas. The interpolation error is localized at discontinuities and can be confounded with the consequence of a bad registration. The SIFT+RANSAC residual has significantly higher values than the IC and mIC residuals. In particular, at the top-right corner of the image (refrigerator edge) the

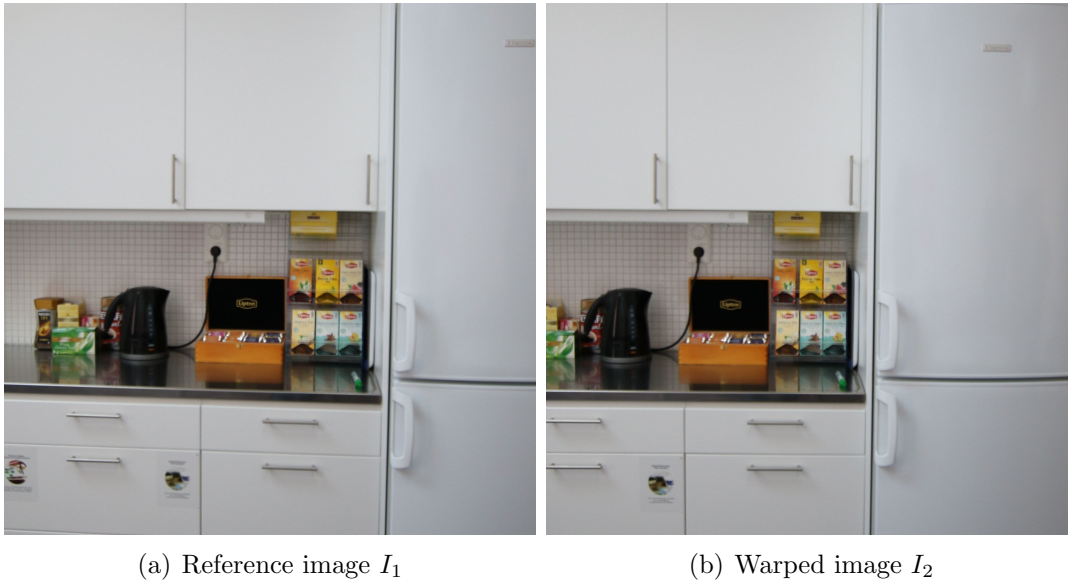


Figure 4: Real data used in Section 4.3.2. The images are taken from the “Lunch Room” sequence of the PASSTA Dataset [13]. Between the acquisitions, the camera was rotated around its optical center so that the images are linked by an homography. The images of size 2048×2048 are stored in JPEG format.

registration is not precise enough. It can be explained by the low density of SIFT keypoints in this zone. The mIC $s_0 = 0$ residual is slightly lower than the IC and mIC $s_0 = 1$ residuals on average but more importantly on the image discontinuities, which is interpreted as a better motion estimation.

By replacing the real motion \mathbf{p}^* by an estimated motion in the end-point error definition in (33), we define the end-point difference between two motions. It allows for a comparison of the estimated motions. The end-point difference fields between the mIC $s_0 = 0$ and the three other methods are shown in Figure 6. The mIC s_0 and SIFT+RANSAC results are considerably different. On average, the end-point difference is greater than 0.5 pixel. They mainly differ at the top-right corner of the image, where the SIFT+RANSAC residual is higher. The mIC $s_0 = 0$ and mIC $s_0 = 1$ mainly differ at the top-left corner of the image. The average end-point difference of 0.15 pixel is not negligible. The finest scale must be used to achieve sufficient precision. The IC and mIC s_0 results are closer but still significantly different. On average, the end-point difference is greater than 0.05 pixel. They mainly differ at the boundaries of the domain, which is a consequence of the different boundary pixels handling but also of the images content.

The estimated motions are computed in respectively 39, 42, 18 and 2 seconds for the SIFT+RANSAC, IC and mIC algorithms. The IC algorithm is slower because at each scale the incremental refinement requires a large amount of iterations. For the mIC algorithm it is the case only for the finest scale. It explains why not using the finest scale divides the computation time by 9.

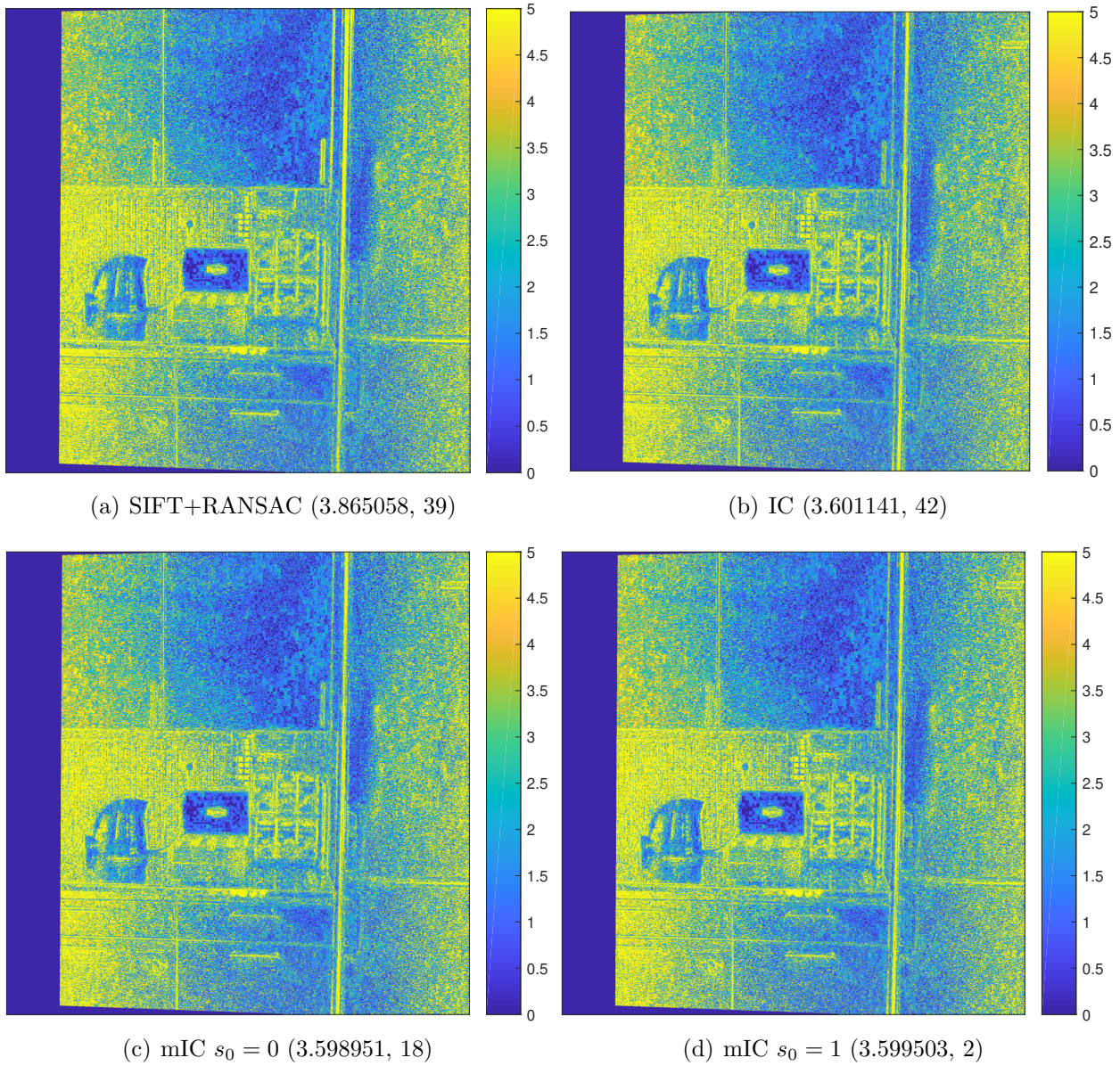


Figure 5: Example of motion estimation on real data. The images correspond to the root mean square over the channels of the residuals $I_1 - I_2(\Psi(\cdot; \mathbf{p}))$, which is computed using bicubic interpolation. The first value between parentheses is the RMSE between I_1 and $I_2(\Psi(\cdot; \mathbf{p}))$. The second value is the computation time expressed in seconds. The residuals are inevitably corrupted by JPEG artifacts, noise and interpolation error. The JPEG artifacts and noise are noticeable on flat areas. The interpolation error is localized at discontinuities and can be confounded with the consequence of a bad registration. The SIFT+RANSAC residual has significantly higher values than the IC and mIC residuals. In particular, at the top-right corner of the image (refrigerator edge) the registration is not precise enough. It can be explained by the low density of SIFT keypoints in this zone. The mIC $s_0 = 0$ residual is slightly lower than the IC and mIC $s_0 = 1$ residuals on average but more importantly on the image discontinuities, which is interpreted as a better motion estimation.

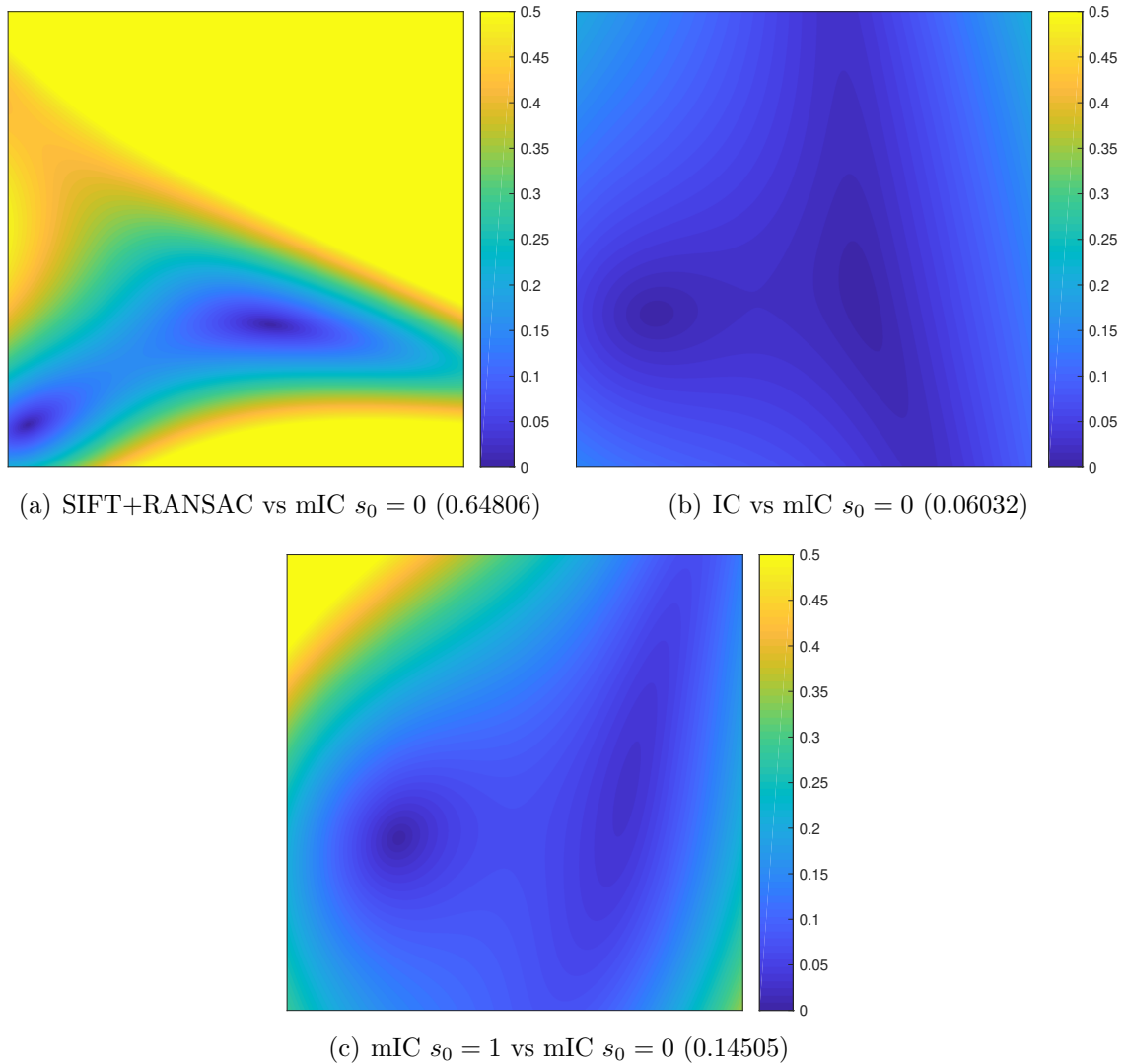


Figure 6: Example of end-point difference fields on real data. The value between parentheses is the average end-point difference. The mIC s_0 and SIFT+RANSAC results are considerably different. On average, the end-point difference is greater than 0.5 pixel. They mainly differ at the top-right corner of the image, where the SIFT+RANSAC residual is higher. The mIC $s_0 = 0$ and mIC $s_0 = 1$ mainly differ at the top-left corner of the image. The average end-point difference of 0.15 pixel is not negligible. The finest scale must be used to achieve sufficient precision. The IC and mIC s_0 results are closer but still significantly different. On average, the end-point difference is greater than 0.05 pixel. They mainly differ at the boundaries of the domain, which is a consequence of the different boundary pixels handling but also of the images content.

5 Conclusion

We detailed the inverse compositional algorithm and proposed to modify it with a correct boundary handling, the grayscale conversion, a more robust gradient estimation applied to a prefiltered image and by skipping scales in the multiscale coarse-to-fine scheme. Discarding boundary pixels is the main source of improvement and must always be done. In general, we recommend to discard boundary pixels with $\delta = 5$ and to use the grayscale conversion and the Farid 5×5 gradient estimator. With these settings, the estimation accuracy is at least improved by a factor 1.3 while the computation time is at least reduced by a factor 2.2 when all the scales are used and by a factor 3.4 when the finest scale is not used. For moderate transformations, the modified algorithm outperforms the classical feature-based methods using the SIFT keypoints and the RANSAC algorithm.

For low quality images, for instance because of noise, using the Farid 5×5 gradient estimator and the grayscale conversion provides the best results. For high quality images, using the central differences gradient estimator without grayscale conversion may provide slightly better results. When efficiency is preferred over accuracy, the grayscale conversion must be used and the finest scale must be skipped.

A Influence of the Color Handling

In Section 4.2.2, we showed that using the grayscale conversion leads to a reduction of the computation time. In addition, the precision of the estimation is similar and the grayscale conversion even provides the best results for large noise values. However, this analysis was conducted using a single image, which is the *RubberWhale* image (see Figure 3). Using the grayscale conversion for other images may deteriorate the precision if the main structures are lost during the conversion. Hopefully, this degenerate case only occurs in images with a specific content.

In this section, we show that the previous analysis remains true on images with classical content. We evaluate the influence of the grayscale conversion using the same experimental setup as in Section 4.1 but with other images. More precisely, we used 12 color images taken from the Middlebury database [4]³. As in Section 4.2.2, boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used.

Results for two images. First, we consider the results for the *Grove2* and *Urban2* images. These two synthetic color images of size 640×480 are shown in Figure 7. The results are presented in Table 12 and Table 13. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.6 to 2.6. The precision results are similar but the grayscale conversion provides the best results for large noise values.

Global results The global results are presented in Table 14. The displayed end-point error and computation time (in ms per image) correspond to the mean over the 12 test sequences of $N_{\text{images}} = 1000$ images. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.6 to 2.3. The precision results are similar but the grayscale conversion provides the best results for large noise values.

³<http://vision.middlebury.edu/flow/data/> (in the “other datasets” part)

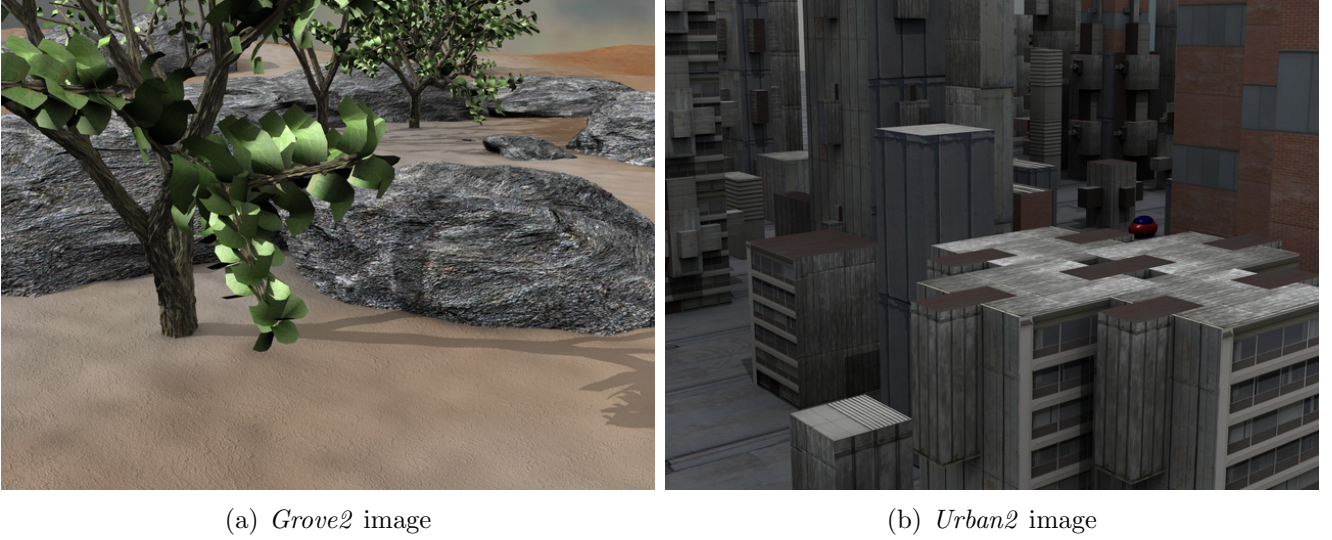


Figure 7: Example of color images taken from the Middlebury database [4]. These two synthetic images are of size 640×480 .

Color handling		L2		Lorentzian	
		Color	Grayscale	Color	Grayscale
$\sigma = 0$	EPE	0.00029	0.00029	0.00030	0.00030
	Time	317	185	402	257
$\sigma = 3$	EPE	0.00089	0.00089	0.00088	0.00089
	Time	330	194	398	265
$\sigma = 5$	EPE	0.00142	0.00136	0.00141	0.00135
	Time	322	193	383	261
$\sigma = 10$	EPE	0.00277	0.00234	0.00279	0.00233
	Time	333	189	370	250
$\sigma = 20$	EPE	0.00802	0.00598	0.00873	0.00603
	Time	407	205	534	268
$\sigma = 30$	EPE	0.01620	0.01077	0.02027	0.01216
	Time	522	233	851	383
$\sigma = 50$	EPE	0.04230	0.02702	0.05478	0.03585
	Time	806	306	1321	664

Table 12: Influence of the color handling using the *Grove2* image. The experimental setup is the same as in Section 4.1. Boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used. The computation time is in ms per image. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.6 to 2.6. For the precision, the results are similar but the grayscale conversion provides the best results for large noise values.

Color handling		L2		Lorentzian	
		Color	Grayscale	Color	Grayscale
$\sigma = 0$	EPE	0.00018	0.00018	0.00017	0.00018
	Time	323	187	410	261
$\sigma = 3$	EPE	0.00164	0.00159	0.00163	0.00159
	Time	324	195	385	264
$\sigma = 5$	EPE	0.00266	0.00244	0.00265	0.00244
	Time	325	191	378	253
$\sigma = 10$	EPE	0.00735	0.00648	0.00744	0.00649
	Time	402	199	471	257
$\sigma = 20$	EPE	0.01986	0.01538	0.02656	0.01756
	Time	626	250	1142	470
$\sigma = 30$	EPE	0.03874	0.02950	0.04797	0.03750
	Time	964	330	1364	715
$\sigma = 50$	EPE	0.08237	0.06832	0.09522	0.07508
	Time	1192	521	1598	1004

Table 13: Influence of the color handling using the *Urban2* image. The experimental setup is the same as in Section 4.1. Boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used. The computation time is in ms per image. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.6 to 2.3. For the precision, the results are similar but the grayscale conversion provides the best results for large noise values.

Color handling		L2		Lorentzian	
		Color	Grayscale	Color	Grayscale
$\sigma = 0$	EPE	0.00012	0.00012	0.00012	0.00011
	Time	295	169	369	229
$\sigma = 3$	EPE	0.00156	0.00177	0.00156	0.00177
	Time	302	174	350	230
$\sigma = 5$	EPE	0.00470	0.00309	0.00470	0.00309
	Time	306	177	357	228
$\sigma = 10$	EPE	0.01082	0.00655	0.01128	0.00666
	Time	377	193	486	261
$\sigma = 20$	EPE	0.02175	0.01712	0.02630	0.01989
	Time	584	260	881	443
$\sigma = 30$	EPE	0.03358	0.03130	0.04188	0.03756
	Time	766	329	1110	596
$\sigma = 50$	EPE	0.06895	0.06518	0.08162	0.07108
	Time	972	435	1370	753

Table 14: Influence of the color handling (global results using 12 images). The experimental setup is the same as in Section 4.1 except that 12 images from the Middlebury database [4] are used. Boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used. The displayed end-point error (EPE in pixels) and computation time (in ms per image) correspond to the mean over the 12 test sequences of $N_{\text{images}} = 1000$ images. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.6 to 2.3. For the precision, the results are similar but the grayscale conversion provides the best results for large noise values.

Acknowledgments

Work partly founded by the Office of Naval research by grant N00014-17-1-2552, ANR-DGA project ANR-12-ASTR-0035, Centre National d'Études Spatiales (CNES, MISS and TOSCA AHA and SMOS-HR Projects), and the European Research Council (advanced grant Twelve Labours n246961).

The authors would also like to thank Prof. Jean-Michel Morel for his support, suggestions, and many fruitful discussions.

Images Credits



Standard test image



Images from the Middlebury database [4]



Images from the "Lunch Room" sequence of the PASSTA Dataset [13]

References

- [1] S. Baker, R. Gross, I. Matthews, and T. Ishikawa. Lucas-Kanade 20 years on: A unifying framework: Part 2. Technical Report CMU-RI-TR-03-01, Pittsburgh, PA, February 2003.
- [2] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I. IEEE, 2001.
- [3] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. <https://doi.org/10.1023/B:VISI.0000011205.11775.fd>.
- [4] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, Mar 2011. <https://doi.org/10.1007/s11263-010-0390-2>.
- [5] H. Farid and E. P. Simoncelli. Differentiation of discrete multidimensional signals. *IEEE Transactions on Image Processing*, 13(4):496–508, 2004. <https://doi.org/10.1109/TIP.2004.823819>.
- [6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. <https://dx.doi.org/10.1145/358669.358692>.
- [7] P. Getreuer. Linear Methods for Image Interpolation. *Image Processing On Line*, 1, 2011. https://dx.doi.org/10.5201/ipol.2011.g_lmii.
- [8] T. Guillemot and J. Delon. Implementation of the Midway Image Equalization. *Image Processing On Line*, 6:114–129, 2016. <https://doi.org/10.5201/ipol.2016.140>.
- [9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. <https://doi.org/10.1023/B:VISI.0000029664.99615.94>.

- [10] W. Lu and Y. Tan. Color filter array demosaicking: new method and performance measures. *IEEE Transactions on Image Processing*, 12(10):1194–1210, 2003. <https://doi.org/10.1109/TIP.2003.816004>.
- [11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. <http://dl.acm.org/citation.cfm?id=1623264.1623280>.
- [12] E. Meinhardt-Llopis, J. Sánchez Pérez, and D. Kondermann. Horn-Schunck Optical Flow with a Multi-Scale Strategy. *Image Processing On Line*, 3:151–172, 2013. <https://doi.org/10.5201/ipol.2013.20>.
- [13] G. Meneghetti, M. Danelljan, M. Felsberg, and K. Nordberg. Image alignment for panorama stitching in sparsely structured environments. In *Image Analysis*, pages 428–439, Cham, 2015. Springer International Publishing. https://dx.doi.org/10.1007/978-3-319-19665-7_36.
- [14] L. Moisan, P. Moulon, and P. Monasse. Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers. *Image Processing On Line*, 2:56–73, 2012. <https://dx.doi.org/10.5201/ipol.2012.mmm-oh>.
- [15] M. Rais. *Fast and accurate image registration. Applications to on-board satellite imaging*. PhD thesis, Université Paris-Saclay, 2016.
- [16] I. Rey Otero and M. Delbracio. Anatomy of the SIFT Method. *Image Processing On Line*, 4:370–396, 2014. <https://doi.org/10.5201/ipol.2014.82>.
- [17] J. Sánchez. The inverse compositional algorithm for parametric registration. *Image Processing On Line*, 6:212–232, 2016. <https://doi.org/10.5201/ipol.2016.153>.
- [18] J. Sánchez Pérez, E. Meinhardt-Llopis, and G. Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3:137–150, 2013. <https://doi.org/10.5201/ipol.2013.26>.
- [19] J. Sánchez Pérez, N. Monzón López, and A. Salgado de la Nuez. Robust Optical Flow Estimation. *Image Processing On Line*, 3:252–270, 2013. <https://doi.org/10.5201/ipol.2013.21>.
- [20] E. P. Simoncelli. Design of multi-dimensional derivative filters. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 790–794. IEEE, 1994.
- [21] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. <https://dx.doi.org/10.1007/978-1-84882-935-0>.
- [22] P. Thevenaz, U. Ruttimann, and M. Unser. A pyramid approach to subpixel registration based on intensity. *IEEE Transactions on Image Processing*, 7(1):27–41, 1998. <https://doi.org/10.1109/83.650848>.