Published in Image Processing On Line on 2014–07–31.
Submitted on 2013–09–26, accepted on 2014–03–20.
ISSN 2105–1232 © 2014 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2014.105

Implementation of the Centroid Method for the Correction of Turbulence

Enric Meinhardt-Llopis[1], Mario Micheli[2]

[1] CMLA, ENS Cachan, France (enric.meinhardt@cmla.ens-cachan.fr)
[2] Department of Mathematics, University of Washington, USA (micheli@uw.edu)

Abstract

The centroid method for the correction of turbulence consists in computing the Karcher-Fréchet mean of the sequence of input images. The direction of deformation between a pair of images is determined by the optical flow. A distinguishing feature of the centroid method is that it can produce useful results from an arbitrarily small set of input images.

Source Code

The source code and a online demo are accessible at the IPOL web page of this article[1].

Keywords: turbulence; centroid

1 Introduction

A common model of turbulence-degraded video is given by random local deformations:

$$I_i(\mathbf{x}) = I(\mathbf{x} + \mathbf{u}_i(\mathbf{x})) \tag{1}$$

where I is a base image without turbulence and $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots$ is a sequence of random vector fields. For practical purposes, the vector fields $\mathbf{u}_i$ are assumed to be smooth and to have a point-wise distribution $\kappa(\mathbf{u})$ of zero mean and finite variance. Analytically, this model can be inverted (i.e., find the image I from the sequence I_i) by different methods. A first method consists in computing the average of all the input images I_i, which converges to the image I blurred by the positive kernel κ, and then de-convolve the average image [6, 7, 9, 16]. A second method consists in inverting the deformation of one of the images I_i to recover the image I [5, 4, 21, 8, 15]. In the first case, one must know exactly the statistical distribution κ of the fields $\mathbf{u}_i$ and have a large enough set of independent images I_i. In the second case, only a single image I_1 is needed, but one must know exactly the field $\mathbf{u}_1$. However, in real problems, neither the distribution κ nor the individual vector fields $\mathbf{u}_i$ are known, so these data must be somehow estimated from the sequence of images I_i. The centroid method

[1] http://dx.doi.org/10.5201/ipol.2014.105

described in the present article is intended to find, or at least approximate, the vector fields $\mathbf{u}_i$ from the sequence of images I_i.

The centroid method for the correction of turbulence was introduced, in a slightly informal way, by Frakes–Monaco–Smith in 2001 [5] and it was formalized recently by Micheli [15]. There are other methods for the correction of turbulence based on optical flow [13]. A distinguishing feature of the centroid method is that it only computes flows between pairs of input images, and it never uses the average of the input images. The main advantage of our method is that it is able to deal with large deformations using a small quantity of images. The main inconvenient is that it is slow for large input sequences (due to the time needed to estimate the vector fields) and that the results are unsatisfactory when there are distortions other than geometric deformations (e.g., local blurs).

2 Description

Optical flow allows to interpolate two images without interpolating their color values (see Figure 1). Let Ω be the image domain (either a rectangle or the whole plane). The action of a vector field $\mathbf{u} : \Omega \to \mathbb{R}^2$ over an image $A : \Omega \to \mathbb{R}$ is defined formally as the *push-forward* of the image A by the mapping $\varphi : \Omega \to \Omega$ given by $\varphi(\mathbf{x}) := \mathbf{x} + \mathbf{u}(\mathbf{x})$. See Figure 2 for an illustration of the push-forward operation. We use the notation

$$A \boxplus \mathbf{u} := A \circ \varphi^{-1}.$$

This operation is only well-defined when the mapping φ is invertible. Far from the boundary of the image domain, this corresponds to the local condition $|D\varphi| > 0$ on the determinant of the Jacobian of φ.

More generally, we can consider the sequence of images

$$A_t := A \boxplus t\mathbf{u} \qquad t \in [0, 1]$$

which interpolates in a continuous way between the images A and $A \boxplus \mathbf{u}$ using only the pixel values of image A. In the particular case where $\mathbf{u} = F_{AB}$ is the optical flow between two images A and B, this means that $A \boxplus \mathbf{u} = B$ and thus the sequence above is a continuous interpolation between the images A and B. For $t = \frac{1}{2}$, the image $A \boxplus \frac{1}{2}\mathbf{u}$ lies at the midpoint, or centroid, between A and B.

The centroid method is the generalization of this construction to an arbitrary quantity of images $I_1, \ldots, I_N$. Instead of computing the linear average image

$$I_{\text{mean}} := \frac{1}{N} \sum_{n=1}^{N} I_n = I_1 + \frac{1}{N} \sum_{n=1}^{N} (I_n - I_1) \tag{2}$$

we transform the image I_1 by *the average of the optical flows*

$$I_{\text{centroid}} := I_1 \boxplus \frac{1}{N} \sum_{n=1}^{N} F_{I_1, I_n} \tag{3}$$

where F_{I_1,I_n} is the optical flow between images I_1 and I_n. Mathematically, this construction corresponds to the Karcher–Fréchet mean [12, 18, 20] on a metric space of images whose geodesics are the curves of the form A_t.

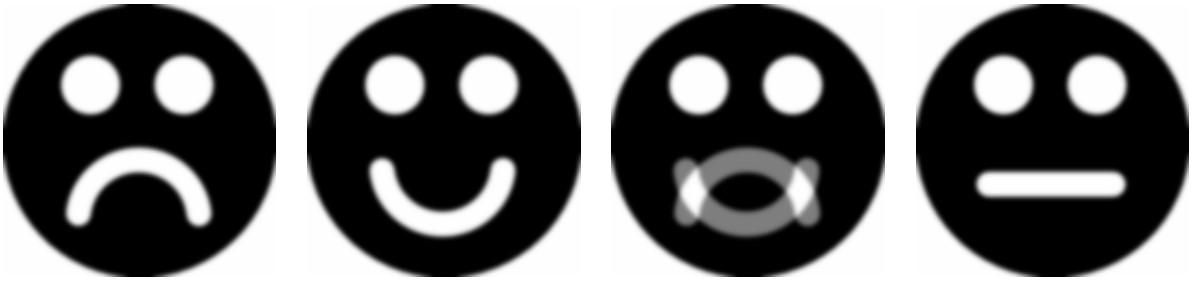
Any general optical flow algorithm can be used in our application; for example Horn–Schunck [10, 14], Lucas–Kanade [2] or TV–L^1 [23, 19]. We have found that this choice is not critical, since all the optical flow estimation methods give essentially correct results over most of the image domain, the

differences being over small areas where the flow is difficult to compute. When taking the average of many of these vector fields, these differences tend to disappear. For simplicity, all the experiments shown here use the Horn–Schunck implementation which is published in IPOL [14], with the same regularization parameter $\alpha = 20$. This choice of $\alpha = 20$ is a hand-tuned compromise with the purpose to find a smooth flow field quickly: larger values of α lead to smoother flow but the method converges very slowly; smaller values of α lead to faster convergence but they are less robust and produce more artifacts. The compromise was reached by looking at the results of the examples presented in this article.

Equation (3) depends on the arbitrary choice of I_1 as reference image. This dependence can be removed by computing the average of the results with each image as reference:

$$I := \frac{1}{N} \sum_{k=1}^{N} \left(I_k \boxplus \frac{1}{N} \sum_{n=1}^{N} F_{I_k, I_n} \right) \tag{4}$$

This new formulation does not depend on any arbitrary parameter, however the running time is prohibitive for a very small gain. A good compromise consists in restricting the above average to a small subset of values of k, for example, one out of every 30 frames, or even a fixed quantity of images. In our implementation, as detailed below, we use a variation of Formula (4) which uses the Weiszfeld median instead of the mean.



A B $A + \frac{1}{2}(B - A)$ $A \boxplus \frac{1}{2} F_{AB}$

Figure 1: Comparison between the linear interpolation of two images and the interpolation by optical flow. Here, F_{AB} is a vector field that represents the optical flow between the images A and B. The notation $\boxplus$ for the *push-forward* is described in the text.

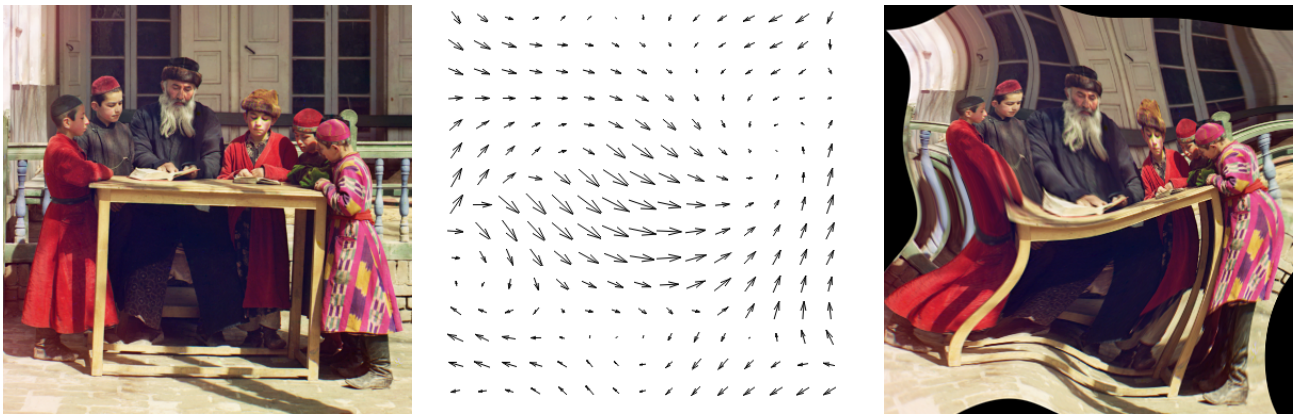
3 Algorithm

The pseudo-code below describes in detail the centroid method for the correction of turbulence. The "main" function is `CentroidCombination` (Algorithm 4) which combines the centroids computed from seven reference frames; this function is a computationally practical version of Formula 4.

The core algorithm is the subroutine `CentroidFromReference` (Algorithm 1), which is simply the transcription of Formula (3). The subroutine `OpticalFlow`(A, B) shall compute a vector field $\mathbf{u}$ such that $A \boxplus \mathbf{u} = B$ approximately. In our case, we use the Horn–Schunck method [10] implemented in IPOL [14], but this can be replaced by any other optical flow technique.

The subroutine `PushForward`$(A, \mathbf{u})$ (Algorithm 2) computes the image $A \boxplus \mathbf{u}$. This computation requires the inversion of a vector field, which is performed by the function `InvertField` (Algorithm 3). The inversion of a vector field $\mathbf{u}$ is defined as a vector field $\mathbf{v}$ such that if

$$\varphi(\mathbf{x}) := \mathbf{x} + \mathbf{u}(\mathbf{x})$$



A $\quad$ $\mathbf{u}$ $\quad$ $A \boxplus \mathbf{u}$

Figure 2: Push-forward of an image by a vector field. Notice that performing this computation requires the inversion of the vector field, which is a nonlinear iterative procedure.

then

$$\varphi^{-1}(\mathbf{x}) = \mathbf{x} + \mathbf{v}(\mathbf{x}).$$

We use a simple algorithm [3] to compute $\mathbf{v}(\mathbf{x})$ which consists in doing 6 recursive applications of the mapping $F(\mathbf{y}) = -\mathbf{u}(\mathbf{x} + \mathbf{y})$, whose fixed point is $\mathbf{y} = \mathbf{v}(\mathbf{x})$. On a neighborhood of $\mathbf{x}$ the Lipschitz constant of this mapping is the operator norm of the Jacobian matrix $D\mathbf{u}(\mathbf{x})$. Thus, for the pixels $\mathbf{x}$ where $\|D\mathbf{u}(\mathbf{x})\| < 1$ holds, the fixed-point iterations converge to the inverse vector field.

The subroutines `PushForward` and `InvertField` need to interpolate images at sub-pixel positions, which is done by the subroutine `InterpolateImageAt`. We propose to use bi-cubic interpolation [11], as implemented by function `BicubicInterpolationImage` on the appendix, but this can be replaced by a higher-order spline.

Algorithm 1: `CentroidFromReference` (implements Equation 3)

Input : Images $I_1, \cdots, I_N$
Input : Parameter $r \in \{1, \ldots, N\}$, index of the reference image
Output: Image I

for $n = 1, \ldots, N$ **do**
    $\mathbf{u}_n \leftarrow$ `OpticalFlow`$(I_r,\ I_n)$
end

$$\mathbf{u} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \mathbf{u}_n$$

$I \leftarrow$ `PushForward`$(I_r,\ \mathbf{u})$

Algorithm 2: `PushForward`

Input : Image I
Input : Vector field $\mathbf{u}$
Output: Image $J = I \boxplus \mathbf{u}$

$\mathbf{v} \leftarrow$ `InvertField`$(\mathbf{u})$
for $\mathbf{x} \in \Omega$ **do**
    $J(\mathbf{x}) \leftarrow$ `InterpolateImageAt`$(I,\ \mathbf{x} + \mathbf{v}(\mathbf{x}))$
end

Algorithm 3: `InvertField`

Input : Vector field $\mathbf{u}$
Output: Vector field $\mathbf{v}$

$\mathbf{v} = 0$
for $i = 1, \ldots, 6$ **do**
  for $\mathbf{x} \in \Omega$ **do**
    $\mathbf{v}(\mathbf{x}) \leftarrow -\texttt{InterpolateImageAt}(\mathbf{u},\ \mathbf{x} + \mathbf{v}(\mathbf{x}))$
  end
end

Algorithm 4: `CentroidCombination` (implements an approximation of Equation 4)

Input : Images $I_1, \cdots, I_N$
Output: Image I

$M \leftarrow 7$
for $i = 1, \ldots, M$ **do**
  $r \leftarrow 1 + \lfloor N/M \rfloor (i-1)$
  $J_i \leftarrow \texttt{CentroidFromReference}(I_1, \ldots, I_n,\ r)$
end
$I \leftarrow \texttt{GeometricMedian}(J_1, \ldots, J_M)$

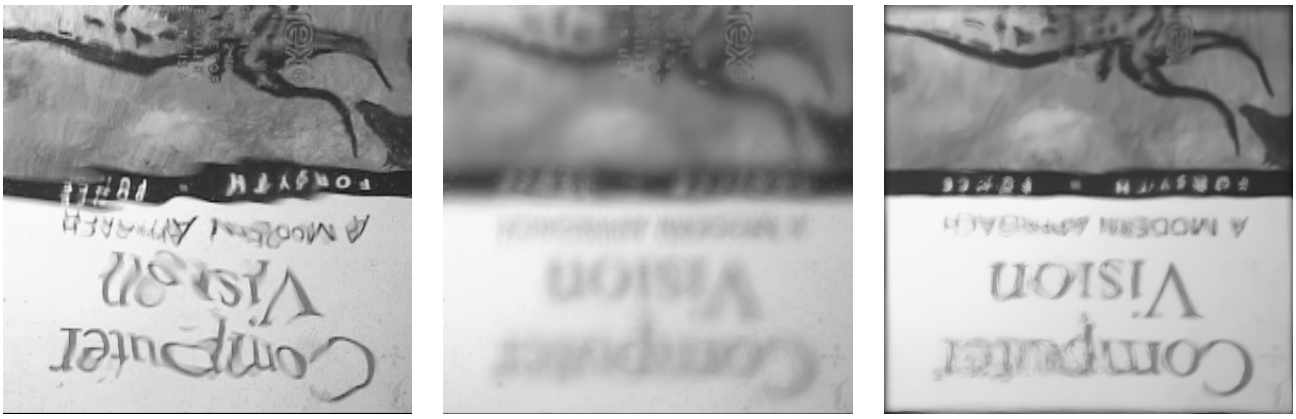
4 Results

It is notoriously difficult to find freely available video sequences with a high degree of turbulence in order to try these algorithms. Since the correction of videos degraded by turbulence has military [13, 15] or sensitive [17] applications, the original sequences used to evaluate published methods are typically not available. In order to evaluate our online implementation we use two video sequences available in the literature from the publications by Efros et al. [4] and Tian et al. [21]. We also use synthetic sequences produced by deforming a color photograph by smooth random fields.

Figures 3 through 5 show the results of the function `CentroidCombination` for these sequences. See the on-line demo associated to this article for a larger collection of experiments. Notice that in general the method is always a clear improvement over the average image. The main advantage of the centroid method is that it can yield useful results with a small amount of input frames (as opposed to deconvolution-based methods, which need a large quantity of input images). In fact, the centroid of two deformed images is well-defined, and it exhibits less deformation than either image.

Appendix: Bicubic interpolation and vector medians

This appendix describes the implementation of some auxiliary functions that are needed in the rest of the algorithm. The bicubic interpolation (algorithms 5, 6 and 7) is used for warping images and fields, and the Weiszfeld vector median [22] (Algorithm 8) can be, optionally, used to combine several centroids as an alternative to Formula 4.



I_1 average centroid

Figure 3: Result of the centroid method for the sequence from reference [4] (used with permission). This is an extract of 400 frames from the original sequence, cropped around the center of the image.

I_1 average centroid

Figure 4: Result of the centroid method for the sequence from reference [21] (used with permission). This is a short sequence of 61 frames with rather large deformations, for which the average has not converged to a uniformly blurry image.

I_1 average centroid

Figure 5: Result of the centroid method over a simulated sequence of 400 frames.

Algorithm 5: `BicubicInterpolationImage` (example of `InterpolateImageAt`)

Input : Image I
Input : Position (x, y) inside the image domain
Output: Value z

$x \leftarrow x - 1$
$y \leftarrow y - 1$
for $i, j = 0, 1, 2, 3$ **do**
    $c_{ij} \leftarrow I\left(\lfloor x \rfloor + i,\ \lfloor y \rfloor + j\right)$
end
$z \leftarrow$ `BicubicInterpolationCell`$(c,\ x - \lfloor x \rfloor,\ y - \lfloor y \rfloor)$

Algorithm 6: `BicubicInterpolationCell`

Input : Sixteen cell values $c_{ij}, \quad i, j = 0, 1, 2, 3$
Input : Position (x, y) inside the cell, $0 \leq x, y < 3$
Output: Value z

$v_0 \leftarrow$ `CubicInterpolationLine`$(c_{00}, c_{01}, c_{02}, c_{03},\ y)$
$v_1 \leftarrow$ `CubicInterpolationLine`$(c_{10}, c_{11}, c_{12}, c_{13},\ y)$
$v_2 \leftarrow$ `CubicInterpolationLine`$(c_{20}, c_{21}, c_{22}, c_{23},\ y)$
$v_3 \leftarrow$ `CubicInterpolationLine`$(c_{30}, c_{31}, c_{32}, c_{33},\ y)$
$z \leftarrow$ `CubicInterpolationLine`$(v_0, v_1, v_2, v_3,\ x)$

Algorithm 7: `CubicInterpolationLine`

Input : Four line values $v_i, \quad i = 0, 1, 2, 3$
Input : Position x, $0 \leq x < 3$
Output: Value y

$$y \leftarrow v_1 + \tfrac{1}{2} \cdot x \cdot \Big(v_2 - v_0 + x \cdot \big(2v_0 - 5v_1 + 4v_2 - v_3 + x \cdot \big(3(v_1 - v_2) + v_3 - v_0 \big) \big) \Big)$$

Algorithm 8: `GeometricMedian` (Weiszfeld's algorithm, [22])

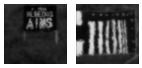
Input : A set of N D-dimensional vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$
Output: The geometric median $\mathbf{y}$

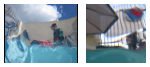
$\varepsilon = 10^{-3}$
$\mathbf{y} \leftarrow \dfrac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$
for $i = 1, \ldots, 5$ **do**
    $$\mathbf{y} \leftarrow \frac{\sum_{n=1}^{N} \mathbf{x}_n / \sqrt{\varepsilon^2 + \|\mathbf{y} - \mathbf{x}_n\|^2}}{\sum_{n=1}^{N} 1 / \sqrt{\varepsilon^2 + \|\mathbf{y} - \mathbf{x}_n\|^2}}$$
end

Image Credits

 Synthetic turbulence by E.Meinhardt-Llopis; base image from the public domain Prokudin-Gorskii Collection

 Van Nevel at the U.S. Naval Air Warfare Center, Weapons Division (China Lake, California), thanks Stanley Osher

 NATO SET156 (ex-SET072) Task Group, thanks Jerome Gilles

 Thanks Y.Y. Schechner and M. Alterman [1]

 A.A. Efros, V.Isler, J. Shi and M. Visontai [4]

 Thanks Y. Tian and S.G. Narasimhan [21]

References

[1] M. Alterman, Y.Y. Schechner, P. Perona, and J. Shamir, *Detecting motion through dynamic refraction*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 245–251. http://dx.doi.org/10.1109/TPAMI.2012.192.

[2] J.-Y. Bouguet, *Pyramidal implementation of the affine Lucas Kanade feature tracker. Description of the algorithm*, Intel Corporation, (2001).

[3] M. Chen, W. Lu, Q. Chen, K.J. Ruchala, and G.H. Olivera, *A simple fixed-point approach to invert a deformation field*, Medical physics, 35 (2008), p. 81. http://dx.doi.org/10.1118/1.2816107.

[4] A. A Efros, V. Isler, J. Shi, and M. Visontai, *Seeing through water*, Advances in Neural Information Processing Systems, 17 (2005), pp. 393–400.

[5] D.H. Frakes, J.W. Monaco, and M.J.T. Smith, *Suppression of atmospheric turbulence in video using an adaptive control grid interpolation approach*, in Proceedings of the (IEEE) International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 3, 2001, pp. 1881–1884. http://dx.doi.org/10.1109/ICASSP.2001.941311.

[6] D.L. Fried, *Probability of getting a lucky short-exposure image through turbulence*, Journal of the Optical Society of America, 68 (1978), pp. 1651–1657. http://dx.doi.org/10.1364/JOSA.68.001651.

[7] P. Getreuer, *Total Variation Deconvolution using Split Bregman*, Image Processing On Line, 2012 (2012). http://dx.doi.org/10.5201/ipol.2012.g-tvdc.

[8] J. Gilles, *Mao-Gilles Stabilization Algorithm*, Image Processing On Line, 2013 (2013), pp. 173–182. http://dx.doi.org/10.5201/ipol.2013.46.

[9] J. Gilles and S. Osher, *Fried deconvolution*, in SPIE Defense, Security and Sensing conference, Baltimore, US, April 2012. http://dx.doi.org/10.1117/12.917234.

[10] B.K.P. Horn and B.G. Schunck, *Determining optical flow*, Artificial intelligence, 17 (1981), pp. 185–203. http://dx.doi.org/10.1016/0004-3702(93)90173-9.

[11] R. Keys, *Cubic convolution interpolation for digital image processing*, IEEE Transactions on Acoustics, Speech and Signal Processing, 29 (1981), pp. 1153–1160. http://dx.doi.org/10.1109/TASSP.1981.1163711.

[12] E. Klassen, A. Srivastava, M. Mio, and S.H. Joshi, *Analysis of planar shapes using geodesic paths on shape spaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 372–383. http://dx.doi.org/10.1109/TPAMI.2004.1262333.

[13] Y. Mao and J. Gilles, *Non rigid geometric distortions correction-application to atmospheric turbulence stabilization*, Inverse Problems and Imaging, 6 (2012), pp. 531–546. http://dx.doi.org/10.3934/ipi.2012.6.531.

[14] E. Meinhardt-Llopis, J. Sanchez, and D. Kondermann, *Horn-Schunck Optical Flow with a Multi-Scale Strategy*, Image Processing On Line, (2013). http://dx.doi.org/10.5201/ipol.2013.20.

[15] M. Micheli, *The centroid method for imaging through turbulence*, arXiv preprint arXiv:1206.3925, (2013).

[16] M. Micheli, Y. Lou, S. Soatto, and A.L. Bertozzi, *A linear systems approach to imaging through turbulence*, Journal of Mathematical Imaging and Vision, (2013). http://dx.doi.org/10.1007/s10851-012-0410-7.

[17] N. Paul, A. de Chillaz, and J.-L. Collette, *On-line restoration for turbulence degraded video in nuclear power plant reactors*, Signal, Image and Video Processing, (2013), pp. 1–10. http://dx.doi.org/10.1007/s11760-013-0497-3.

[18] X. Pennec, *Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements*, Journal of Mathematical Imaging and Vision, 25 (2006), pp. 127–154. http://dx.doi.org/10.1007/s10851-006-6228-4.

[19] J. Sanchez, E. Meinhardt-Llopis, and G. Facciolo, *TV-L1 Optical Flow Estimation*, Image Processing On Line, (2013). http://dx.doi.org/10.5201/ipol.2013.26.

[20] N. Thorstensen, F. Segonne, and R. Keriven, *Pre-Image as Karcher Mean using Diffusion Maps: Application to shape and image denoising*, in Scale Space and Variational Methods in Computer Vision, Springer, 2009, pp. 721–732. http://dx.doi.org/10.1007/978-3-642-02256-2_60.

[21] Y. Tian and S.G. Narasimhan, *Seeing through water: Image restoration using model-based tracking*, in IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 2303–2310. http://dx.doi.org/10.1109/ICCV.2009.5459440.

[22] E. Weiszfeld, *Sur le point pour lequel la somme des distances de n points donnés est minimum*, Tohoku Mathematics Journal, 43 (1937), pp. 355–386.

[23] C. Zach, T. Pock, and H. Bischof, *A duality based approach for realtime TV-L1 optical flow*, in Pattern Recognition, Springer, 2007, pp. 214–223. http://dx.doi.org/10.1007/978-3-540-74936-3_22.