2015/06/16 v0.5.1 IPOL article class

Published in Image Processing On Line on 2013–07–19.
Submitted on 2012–10–08, accepted on 2013–07–02.
ISSN 2105–1232 © 2013 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.2013.46

Mao-Gilles Algorithm for Turbulence Stabilization

Jérôme Gilles[1]

[1] Department of Mathematics, UCLA, USA (jegilles@math.ucla.edu)

Communicated by Luis Álvarez *Demo edited by* Yohann Tendero

Abstract

The Mao-Gilles stabilization algorithm was designed to compensate the non-rigid deformations due to atmospheric turbulence. Given a sequence of frames affected by atmospheric turbulence, the algorithm uses a variational model combining optical flow and regularization to characterize the static observed scene. The optimization problem is solved by Bregman Iteration and the operator splitting method. The algorithm is simple, efficient, and can be easily generalized for different scenarios involving non-rigid deformations (for instance, in cardiac imagery where some deformations are present due to the heart beat).

Source Code

A C implementation of this algorithm is provided. This version uses a $TV - L^1$ optical flow algorithm [3] and the Rudin-Osher-Fatemi [2] regularization. The source code, the code documentation, and the online demo are accessible at the IPOL web page of this article[1].

Keywords: stabilization, non-rigid, Bregman, operator splitting, turbulence restauration

1 Introduction

In long range imaging, the quality of the observed image is affected by the atmospheric turbulence, see figure 2 for examples of geometrically deformed images. Frakes et al. [4] propose to model these defects as the combination of time-dependent geometrical distortions and some blur. Mao and Gilles [1] focus on the compensation of the geometrical distortions. In this paper, we propose an implementation of this algorithm.

We denote the observed image sequence by $\{f_i\}_{i=1,\dots,N}$ (N is the number of frames) and the true image that needs to be reconstructed by u. We assume

$$f_i(x) = u(\phi_i(x)) + \text{noise}, \quad \forall i \tag{1}$$

where ϕ_i corresponds to the geometric deformation on the i-th frame (note that the ϕ_i are the deformations between the true image and the observed frame i and not the continuous movement

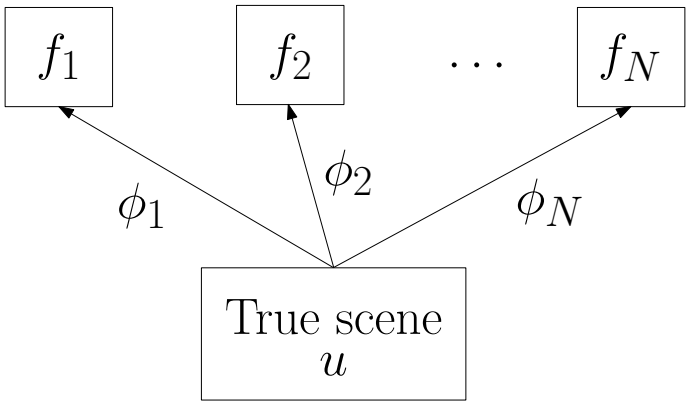

Figure 1: The deformation model

flow from frame to frame, see figure 1). Mao and Gilles propose to invert the geometrical deformations by solving problem (2).

$$\min_{u,\phi_i} J(u) \quad \text{s.t.} \quad f_i = \Phi_i u, \quad \forall i \tag{2}$$

where $J(u)$ is a regularization term applying on u like, for example, the total variation (TV) [2], the nonlocal TV [5] or some sparsity constraint in a basis or frame representation [6]. The notation $\Phi_i u$ stands for the fact that the deformation map acts linearly on u ($\Phi_i(u_1+u_2) = u_1(\phi_i(x)) + u_2(\phi_i(x))$). The choice of the regularizer basically depends on the type of scene we want to restore. For most of the natural images, total variation is enough to recover a visually "good" picture. But, for instance, in the case of images which contain a lot of textures and details, the nonlocal TV should provide better results but with an increasing of the computational cost. In this paper, we only address the case when $J(u)$ is the total variation. This formulation returns us to a well known problem described in the next section.

2 Algorithm

An optimization problem like (2) can be solved by an alternating optimization method, i.e. optimizing over different variables alternatively. First, let us assume that u is fixed (the choice of the initial u will be discussed later), then the optimal ϕ_i can be estimated by (1) via certain optical flow algorithms (in this paper we will use the $TV-L^1$ algorithm implementation by Sánchez et al. [3]). On the other hand, for fixed $\{\phi_i\}$ model (2) is equivalent to a constrained problem which have the general form (if A is a compact linear operator defined from L^2 to L^2, see the paper by Osher et al. [7] for more details):

$$\min_{u} J(u) \quad \text{s.t.} \quad f = Au, \tag{3}$$

and can be efficiently solved by the use of Bregman Iteration (4) (see [7] for details).

$$\begin{cases} u^k = \arg\min_u J(u) + \frac{\lambda}{2}\|Au - f^k\|^2 \\ f^{k+1} = f^k + f - Au^k, \end{cases} \tag{4}$$

[1] https://doi.org/10.5201/ipol.2013.46

where λ is a regularization parameter. The first step in (4) is an unconstrained problem and can be solved by the forward-backward operating splitting method [8].

$$\begin{cases} v \leftarrow u - \delta A^\top (Au - f) \\ u \leftarrow \arg\min_u J(u) + \frac{\lambda}{2\delta}\|u - v\|^2. \end{cases} \tag{5}$$

The first line, the forward step, is the gradient descent of $\|Au-f\|^2$ with the time step δ. The second line

$$u \leftarrow \arg\min_u J(u) + \frac{\lambda}{2\delta}\|u - v\|^2, \tag{6}$$

is called the backward step and can be solved efficiently for various forms of $J(u)$. For example, if we choose the total variation as the regularization term then (6) is the standard ROF model [2]. We apply this general optimization strategy to solve (2) by setting $\|Au - f\|_2^2 = \sum_i \|\Phi_i u - f_i\|_2^2$ and combining the updating step for Φ_i into the Bregman updating loop. Then the whole algorithm to solve (2) is given in algorithm 1.

Algorithm 1: Mao-Gilles stabilization algorithm.

Inputs: Sequence of frames f_i, parameters $\lambda, \delta, N_{bregman}, N_{splitting}$;
Initialize: Start from some initial guess for u. Let $\tilde{f}_i = f_i$;
for $N_{bregman}$ *iterations* **do**
    Estimate each Φ_i which maps u onto f_i from (1) via an optical flow scheme;
    for $N_{splitting}$ *iterations* **do**
        $v \leftarrow u - \delta \sum_i \Phi_i^\top(\Phi_i u - \tilde{f}_i)$;
        $u \leftarrow \arg\min_u J(u) + \frac{\lambda}{2\delta}\|u - v\|^2$;
    end
    $\tilde{f}_i \leftarrow \tilde{f}_i + f_i - \Phi_i u$;
end
Output: u

3 Numerical Implementation

3.1 Initial Guess for u

In the initialization step of algorithm 1, we need an initial guess u. The simplest natural guess is the temporal average of the input sequence:

$$u_{init} = \frac{1}{N}\sum_{i=1}^{N} f_i. \tag{7}$$

This initial guess has less geometrical distortions but is very blurry.

3.2 Optical Flow

To estimate the deformation maps Φ_i we use an optical flow algorithm. If the operator Φ_i is the operator which maps a function v to a function w ($w = \Phi_i v$), its adjoint, $\Phi_i^\top$, is the operator which maps w to v ($v = \Phi_i^\top w$). By using the optical flow formalism, we have $w(x) = (\Phi_i v)(x) = v(\phi_i(x))$

where $\phi_i(x) = x + \varphi_i(x)$ and φ_i is the optical flow (vector field). This operation can be interpreted as "we map the pixel at position $x+\varphi_i(x)$ in v to the pixel at position x in w". Hence the adjoint operator consists to map the pixel at position x in w to the pixel at position $x + \varphi_i(x)$ in v or equivalently we map the pixel at position $x - \varphi_i(x)$ in w to a pixel at position x in v: $v(x) = \left(\Phi_i^\top w\right)(x) = w(\phi_i^\top(x))$ where $\phi_i^\top(x) = x - \varphi_i(x)$. In practice the grids on both images do not coincide then a bilinear interpolation is used to get the vectors on the regular grid.
Mao and Gilles [1] show that the choice of the optical flow algorithm is not crucial (they get quasi identical results both by using the classic Lukas-Kanade algorithm [10], which is pretty fast but not precise, and the Black-Anandan algorithm [9], which is more precise but slower). In the proposed implementation of this paper, we use the $TV-L^1$ optical flow algorithm currently available in IPOL[2] and developed by Sánchez et al. [3]. In this paper, we use the default parameters suggested by these authors.

3.3 Rudin-Osher-Fatemi Regularization

In the proposed implementation, we use the TV-based Rudin-Osher-Fatemi (ROF) algorithm to regularize the image. While we use the Bregman iterations to solve the overall problem, this TV regularization is implemented with the help of Chambolle's method [11] but some other methods like the split Bregman approach can be used [12, 13]. To solve a problem of type (8),

$$\hat{u} = \arg_u \min J(u) + \frac{\lambda}{2}\|u - v\|_2^2, \tag{8}$$

where v is the original image, $J(u)$ is the total variation of u (defined as $J(u) = \int |\nabla u|$) and $\hat{u}$ the regularized image, Chambolle shows that the solution of (8) can be written as $\hat{u} = f - P_{G_{1/\lambda}}(v)$ where $P_{G_\mu}(v)$ is evaluated as shown in Proposition. 1.

Proposition 1 *If $\tau < \frac{1}{8}$ (in the proposed implementation, we set $\tau = 0.12$ to fulfill this condition),$\mu > 0$ then $\mu \mathrm{div}\,(p^n)$ converges to $P_{G_\mu}(v)$ when $n \to +\infty$ where*

$$p_{m,n}^{k+1} = \frac{p_{m,n}^k + \tau\left(\nabla\left(\mathrm{div}\,(p^k) - \frac{v}{\mu}\right)\right)_{m,n}}{1 + \tau\left|\left(\nabla\left(\mathrm{div}\,(p^k) - \frac{v}{\mu}\right)\right)_{m,n}\right|}. \tag{9}$$

The discrete gradient (∇u) and divergence (div p where $p = (p^1, p^2)$) operators are respectively defined by

$$(\nabla u)_{m,n} = \left((\nabla u)_{m,n}^1, (\nabla u)_{m,n}^2\right), \tag{10}$$

where $\forall (m,n) \in [0,\ldots,M-1] \times [0,\ldots,N-1]$ (i.e. the input image is of size $M \times N$),

$$(\nabla u)_{m,n}^1 = \begin{cases} u_{m+1,n} - u_{m,n} & \text{if} \quad m < M-1 \\ 0 & \text{if} \quad m = M-1 \end{cases} \tag{11}$$

$$(\nabla u)_{m,n}^2 = \begin{cases} u_{m,n+1} - u_{m,n} & \text{if} \quad n < N-1 \\ 0 & \text{if} \quad n = N-1, \end{cases} \tag{12}$$

and

$$(\mathrm{div}\ p)_{m,n} = \begin{cases} p_{m,n}^1 - p_{m-1,n}^1 & \text{if} \quad 0 < m < M-1 \\ p_{m,n}^1 & \text{if} \quad m = 0 \\ -p_{m-1,n}^1 & \text{if} \quad m = M-1 \end{cases} + \begin{cases} p_{m,n}^2 - p_{m,n-1}^2 & \text{if} \quad 0 < n < N-1 \\ p_{m,n}^2 & \text{if} \quad n = 0 \\ -p_{m,n-1}^2 & \text{if} \quad n = N-1. \end{cases} \tag{13}$$

[2]Image Processing Online: http://www.ipol.im/

Moreover, in the proposed code, the stopping criteria is based on two tests: if we reached a prescribed number of iterations (set to 20) and if the L^2 error between two iterations is smaller than a value ϵ (set to 0.001).

3.4 Main and Inner Loop

Algorithm 1 has two loops which are theoretically stopped when their respective convergence criteria are reached. In practice, Mao and Gilles [1] note that the algorithm gives good results even if some prescribed number of iterations are used. These numbers of iterations are called $N_{bregman}$ and $N_{splitting}$ corresponding to the main Bregman loop and the inner Splitting loop, respectively. These parameters are inputs parameters of the algorithm. In the proposed source code, we set $N_{bregman} = 4$ and $N_{splitting} = 5$.

3.5 Total vs. Shifted Algorithms

The proposed implementation provides two executable programs: *MaoGilles* and *ShiftMaoGilles*. The first one is the direct implementation of the previously described Algorithm 1. From a sequence of N images it gives a single restored image. However, generally, the goal of a stabilization algorithm is to provide a stabilized sequence and not only a single image. This is the purpose of the *ShiftMaoGilles* program. Basically, we add one more parameter: a temporal window size, N_T; then we perform the initial Mao-Gilles algorithm on each shifted temporal window. This algorithm is summarized in algorithm 2. The reader should keep in mind that the provided source code for the *ShiftMaoGilles* program is not optimized in terms of computational cost as we recursively apply the initial Mao-Gilles code. For example, some improvement can be made if the restoration at the temporal window $l + 1$ uses the restored image obtained from the temporal window l. Moreover some optical flow computation can be reused. The implementation of these optimizations falls beyond the scope of this paper, as it needs to split the optical flow algorithm into more basics functions that would need to be rearranged and adapted.

Algorithm 2: Shifted Mao-Gilles stabilization algorithm.

Inputs: Sequence of frames f_i, parameters $\lambda, \delta, N_{bregman}, N_{splitting}$ and the temporal window size $N_T < N$;
Initialization: $l = 0$;
while $l < N - N_T$ **do**
    Extract the subsequence $\{f_i\}^l = \{f_i\}_{l \leq i < l+N_T}$;
    Apply the Mao-Gilles stabilization algorithm to $\{f_i\}^l$;
    Save the corresponding output u^l;
    $l = l + 1$;
end
Output: the sequence of frames $\{u^l\}$

4 Results

Concerning the parameters λ and δ, while there are no theoretical results giving any clues about their choice, the experience shows that if λ is too large (typically ≥ 1), the algorithm diverges. However λ must not be too small otherwise it allows larger error between f_i and $\Phi_i u$ which is not what we expect. Experimentally, we can check that δ must be chosen in the range $[0.05; 1]$. If $\delta > 1$,

the algorithm diverges while if $\delta < 0.05$, the algorithm does not retrieve sharp edges. In all the experiments, we use the same parameters: $\lambda = 0.1, \delta = 0.5$.

The input sequences are shown on figure 2 as well as their "ground truth" in figure 3. We call them, from top to bottom, *bars* (it is a real sequence acquired through atmospheric turbulence, note that the "wire" which can be seen on the images is due to the camera design and is not affected by the turbulence phenomena), *lena* and *poème*, respectively. In the last two ones, the deformations were synthetically generated by the *rippling* filter available in the software *The Gimp*. Table 1 indicates the number of frames for each sequence, their size and the processing time taken to get the result (experiments made on a core-i5 at 2.5GHz with 8Gb of memory).

Figure 2: Example of deformed images. The first row shows real observations through atmospheric turbulence (*bars* sequence) while the other rows show simulated deformations (*lena* and *poème* sequences).

The results of the total Mao-Gilles stabilization applied on the whole sequences are given in figure 4, as well as the corresponding temporal average of each input sequences. We can see that the structure distortions present in the original frames are rectified. Moreover, our method provides sharper results and more details (see at the textures in Lena's hat) compared to the temporal averages of input frames.

Figure 5 presents the results obtained with the shifted Mao-Gilles stabilization with $N_T = 20$ for the *bars* sequence, $N_T = 20$ for the *lena* sequence and $N_T = 30$ for the *poème* sequence, respectively.

Figure 3: Ground Truth for each sequence. The *bars* one is a schematic drawing which was painted on a real $2m \times 1m$ board.

Sequence	Number of frames	Size	Computing time (in s)
bars	10	256×256	8
lena	30	256×256	21
poème	50	576×330	112

Table 1: Computational times for each test sequences.

In these experiments, we can see that the deformation amplitudes are significantly reduced. Obviously, the choice of N_T will affect the stabilization quality. Indeed, if N_T is too small then the output sequence has less deformation than the original frames but still contain some movement. If N_T is large, the output sequence is really well stabilized but the computational cost becomes important. While we originally developed this method to compensate turbulence deformations, it can be useful for many other kind of applications. For example in medical imaging, sequences showing the heart can be acquired. In such a case, the heart beats generate non-rigid deformations along the sequence. Our method should be able to provide a stabilized image of the heart.

Acknowledgments

The author wants to thank the members of the NATO SET156 (ex-SET072) Task Group for the opportunity of using the data collected by the group during the 2005 New Mexicos field trials. The author also wants to thanks the reviewers for their suggestions which permit to improve this submission.

Image Credits


Standard test images.


NATO-SET156.

Figure 4: Stabilized (left) and temporal average (right) images obtained from the *bars, lena* and *poème* sequences, respectively.

Figure 5: Three consecutive stabilized frames computed from the *bars*, *lena* and *poème* sequences.

References

[1] Y. Mao and J. Gilles, *Non rigid geometric distortions correction - Application to atmospheric turbulence stabilization*, Journal of Inverse Problems and Imaging, vol. 6, no. 3, pp. 531–546 , August 2012. http://dx.doi.org/10.3934/ipi.2012.6.531. Preprint.[3]

[2] L. Rudin and S. Osher and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, vol. 60, pp. 259–268, 1992. http://dx.doi.org/10.1016/0167-2789(92)90242-F

[3] J. Sánchez Pérez and E. Meinhardt-Llopis and G. Facciolo, *$TV - L^1$ optical flow estimation*, Image Processing Online Preprint, 2012. http://www.ipol.im/pub/algo/smf_tvl1_optical_flow_estimation/

[4] D. Frakes, J. Monaco and M. Smith, *Suppression of atmospheric turbulence in video using an adaptive control grid interpolation approach*, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. http://dx.doi.org/10.1109/ICASSP.2001.941311

[5] G. Gilboa and S. Osher, *Nonlocal operators with applications to image processing*, Multiscale Modeling and Simulation, vol. 7, pp. 1005–1028, 2008. http://dx.doi.org/10.1137/070698592

[6] J.F. Cai, S. Osher and Z. Shen, *Split Bregman methods and frame based image restoration*, Multiscale Modeling and Simulation, vol. 8, pp. 337–369, 2009. http://dx.doi.org/10.1137/090753504

[7] S. Osher, M. Burger, D. Goldfarb, J. Xu and W. Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Modeling and Simulation, vol. 4, pp. 460–489, 2005. http://dx.doi.org/10.1137/040605412

[8] P.L. Combettes and V.R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Modeling and Simulation, vol. 4, pp. 1168–1200, 2005. http://dx.doi.org/10.1137/050626090

[9] M.J. Black and P. Anandan, *The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields*, Computer Vision and Image Understanding, vol. 63, pp. 75–104, 1996. http://dx.doi.org/10.1006/cviu.1996.0006

[10] J.Y. Bouguet, *Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm*, Intel Corporation Microprocessor Research Labs, 2000 Tech. Report[4].

[11] A. Chambolle, *An algorithm for total variation minimization and applications*, Journal of Mathematical Imaging and Vision, vol. 20, No. 1-2, pp. 89–97, 2004 http://dx.doi.org/10.1023/B:JMIV.0000011325.36760.1e

[12] P. Getreuer, *Rudin-Osher-Fatemi total variation denoising using split Bregman*, Image Processing On Line, 2012 http://dx.doi.org/10.5201/ipol.2012.g-tvd

[13] P. Getreuer, *Total variation deconvolution using split Bregman*, Image Processing On Line, 2012 http://dx.doi.org/10.5201/ipol.2012.g-tvdc

[3] ftp://ftp.math.ucla.edu/pub/camreport/cam10-86.pdf
[4] http://robots.stanford.edu/cs223b04/algo_tracking.pdf