Mao-Gilles Algorithm for Turbulence Stabilization

Jérôme Gilles¹

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 article¹.

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

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

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

(1)

where $\phi_i$ corresponds to the geometric deformation on the $i$-th frame (note that the $\phi_i$ are the deformations between the true image and the observed frame $i$ and not the continuous movement...
Mao-Gilles Algorithm for Turbulence Stabilization

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

\[
\min_{u, \phi_i} J(u) \ \text{s.t.} \ f_i = \Phi_i u, \ \forall i
\]

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 \( \phi_i \) 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):

\[
\min_u J(u) \ \text{s.t.} \ f = Au,
\]

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

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

\[\text{https://doi.org/10.5201/ipol.2013.46}\]
where \( \lambda \) 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].

\[
\begin{aligned}
& v \leftarrow u - \delta A^\top (Au - f) \\
& u \leftarrow \arg \min_u J(u) + \frac{\lambda}{2\delta} \|u - v\|^2 .
\end{aligned}
\] (5)

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

\[
\begin{aligned}
& u \leftarrow \arg \min_u J(u) + \frac{\lambda}{2\delta} \|u - v\|^2 ,
\end{aligned}
\] (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 \( \|Au - f\|^2 = \sum_i \|\Phi_i u - f_i\|^2_2 \) and combining the updating step for \( \Phi_i \) into the Bregman updating loop. Then the whole algorithm to solve (2) is given in algorithm 1.

**Algorithm 1:** Mao-Gilles stabilization algorithm.

1. Inputs: Sequence of frames \( f_i \), parameters \( \lambda, \delta, N_{\text{bregman}}, N_{\text{splitting}} \);

2. Initialize: Start from some initial guess for \( u \). Let \( \tilde{f}_i = f_i \);

3. for \( N_{\text{bregman}} \) iterations do

4. Estimate each \( \Phi_i \) which maps \( u \) onto \( f_i \) from (1) via an optical flow scheme;

5. for \( N_{\text{splitting}} \) iterations do

6. \( v \leftarrow u - \delta \sum_i \Phi_i^\top (\Phi_i u - \tilde{f}_i) \);

7. \( u \leftarrow \arg \min_u J(u) + \frac{\lambda}{2\delta} \|u - v\|^2 \);

8. end

9. \( \tilde{f}_i \leftarrow \tilde{f}_i + f_i - \Phi_i u \);

10. end

11. 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:

\[
\begin{aligned}
& u_{\text{init}} = \frac{1}{N} \sum_{i=1}^N f_i .
\end{aligned}
\] (7)

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

#### 3.2 Optical Flow

To estimate the deformation maps \( \Phi_i \) we use an optical flow algorithm. If the operator \( \Phi_i \) 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 $\varphi_i$ 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 $v$: $v(x) = (\Phi_i^T w)(x) = w(\phi_i^T(x))$ where $\phi_i^T(x) = x - \varphi_i(x)$. 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),

$$\hat{u} = \arg_{u} \min J(u) + \frac{\lambda}{2} \|u - v\|_2^2; \quad (8)$$

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 $\mu \text{div}(p^n)$ converges to $P_{G_\mu}(v)$ when $n \to +\infty$ where

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

The discrete gradient ($\nabla u$) and divergence ($\text{div} p$ where $p = (p^1, p^2)$) operators are respectively defined by

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

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

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

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

and

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

\(^2\)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 $\epsilon$ (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 $N_{\text{bregman}}$ and $N_{\text{splitting}}$ 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 $N_{\text{bregman}} = 4$ and $N_{\text{splitting}} = 5$.

### 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.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Inputs: Sequence of frames $f_i$, parameters $\lambda, \delta, N_{\text{bregman}}, N_{\text{splitting}}$ and the temporal window size $N_T &lt; N$;</td>
</tr>
<tr>
<td>2</td>
<td>Initialization: $l = 0$;</td>
</tr>
<tr>
<td>3</td>
<td>while $l &lt; N - N_T$ do</td>
</tr>
<tr>
<td>4</td>
<td>Extract the subsequence ${f_i}^l = {f_i}_{l \leq i &lt; l + N_T}$;</td>
</tr>
<tr>
<td>5</td>
<td>Apply the Mao-Gilles stabilization algorithm to ${f_i}^l$;</td>
</tr>
<tr>
<td>6</td>
<td>Save the corresponding output $u^l$;</td>
</tr>
<tr>
<td>7</td>
<td>$l = l + 1$;</td>
</tr>
<tr>
<td>8</td>
<td>end</td>
</tr>
<tr>
<td>9</td>
<td>Output: the sequence of frames ${u^l}$</td>
</tr>
</tbody>
</table>

### 4 Results

Concerning the parameters $\lambda$ and $\delta$, while there are no theoretical results giving any clues about their choice, the experience shows that if $\lambda$ is too large (typically $\geq 1$), the algorithm diverges. However $\lambda$ must not be too small otherwise it allows larger error between $f_i$ and $\Phi_iu$ which is not what we expect. Experimentally, we can check that $\delta$ 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.
Des voix parlaient; pour qui ? Pour l'espace sans bornes,
Pour le recueillement des solitudes mornes,
Pour l'oreille, partout éparse, du désert ;
Nulle part, dans la plaine où le regard se perd,
On ne voyait marcher la foule aux bruits sans nombre,
Mais on sentait que l'homme écoutait dans cette ombre.
Qui donc parlait ? C'étaient des monuments pensifs,
Debout sur l'onde humaine ainsi que des récifs,
Calmes, et chacun d'eux semblait un personnage
Vivant, et se rendant lui-même témoignage.
Nulle rumeur n'osait à ces voix se mêler,
Et le vent se taisait pour les laisser parler,
Et le flot apaisait ses mystérieux râles.
Un soleil vague au loin dorait les frontons pâles.
Les astres commençaient à se faire entrevoir
Dans l'assombrissement religieux

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

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Number of frames</th>
<th>Size</th>
<th>Computing time (in s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bars</td>
<td>10</td>
<td>256 × 256</td>
<td>8</td>
</tr>
<tr>
<td>lena</td>
<td>30</td>
<td>256 × 256</td>
<td>21</td>
</tr>
<tr>
<td>poème</td>
<td>50</td>
<td>576 × 330</td>
<td>112</td>
</tr>
</tbody>
</table>

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 Mexico 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.
Mao-Gilles Algorithm for Turbulence Stabilization

References


207