FAST TV-L_1 OPTICAL FLOW FOR INTERACTIVITY

Emmanuel d'Angelo, Johan Paratte, Gilles Puy, Pierre Vandergheynst

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory

ABSTRACT

Vision is a natural tool for human-computer interaction, since it provides visual feedback to the user and mimics some human behaviors. It requires however the fast and robust computation of motion primitives, which remains a difficult problem. In this work, we propose to apply some recent mathematical results about convex optimization to the TV-L_1 optical flow problem. At the cost of a small smoothing of the Total Variation (TV), the convergence speed of the numerical scheme is improved, leading to earlier termination. Furthermore, we successfully implement our algorithm on GPU for realtime performance using the OpenCL framework.We demonstrate the potential of our optical flow by using it as primary sensor in a remotely controlled image browsing software.

Index Terms— Optical flow, two, three, four, OpenCL

1. INTRODUCTION

Since its introduction by Horn and Schunck [1], the problem of *variational optical flow* estimation has been an active area of research. In this approach, the under-determination of the motion estimation problem is accounted for by enforcing constraints on the flow regularity. Then, the objective function to be minimized becomes the sum of an error (or registration) term and a flow regularity term. If I_0 and I_1 are two images defined on a domain $\Omega \subset \mathbb{R}^2$, then the desired optical flow $\mathbf{u} = (u_1, u_2)^T$ is the solution of the problem:

$$\underset{\mathbf{u}}{\operatorname{argmin}} \underbrace{\sum_{x\in\Omega} \lambda\rho\left(I_0(x) - I_1(x+\mathbf{u}(x))\right)}_{\text{registration error}} + \underbrace{\sum_{x\in\Omega} \kappa\left(\nabla\mathbf{u}(x)\right)}_{\text{regularity}}. \quad (1)$$

While Horn and Schunck used the quadratic registration error ($\rho(z) = z^2$) and the Tikhonov regularization ($\kappa(\nabla\mathbf{u}) = \|\nabla\mathbf{u}\|^2$), subsequent work was directed towards replacing these constraints by more robust terms.

Indeed, discontinuities in the optical flow field should be allowed along the boundaries of objects, since they may have different motion than their surroundings. Many different techniques were applied to deal with this issue. For example, Black and Anandan introduced in [2] the use of robust statistics to replace the quadratic expressions of $\rho(\cdot)$ and $\kappa(\cdot)$, while Pérez and Mémin [3] proposed to solve a problem coupling optical flow and image segmentation. However, these formulations lead to non-convex optimization problems, that are non trivial to solve.

In order to have better convergence properties, Deriche *et al.* [4] introduced an *anisotropic diffusion*-based differential algorithm. Later, [5] and [6] popularized the use of the L_1 norm for both $\rho(\cdot)$ and $\kappa(\cdot)$. In this case the regularity constraint becomes the Total Variation (TV) norm of the optical flow: $\|\mathbf{u}\|_{\text{TV}} = |\nabla\mathbf{u}| = \kappa(\nabla u)$. Eq. (1) then becomes:

$$\underset{\mathbf{u}}{\operatorname{argmin}} \sum_{x\in\Omega} \lambda|I_0(x) - I_1(x+\mathbf{u}(x))| + \sum_{x\in\Omega} \|\mathbf{u}(x)\|_{\text{TV}}, \quad (2)$$

hence the name of TV-L_1 optical flow.

While the objective function in Eq. (2) is convex, it is not differentiable near 0 because of the TV norm. To overcome this difficulty, Zach *et al.* introduced in [7] an auxiliary variable $\mathbf{v}$ such that the problem becomes:

$$\underset{\mathbf{u},\mathbf{v}}{\operatorname{argmin}} \sum_{x\in\Omega} \lambda|I_0(x) - I_1(x+\mathbf{u}(x))| + \frac{1}{2\theta}(\mathbf{u}-\mathbf{v})^2 + \sum_{x\in\Omega} \|\mathbf{u}(x)\|_{\text{TV}}, \quad (3)$$

where θ is a (small) additional parameter. By *splitting* this problem, one gets two smooth convex subproblems:

1. solving for $\mathbf{v}$ yields an L_2-L_1 formulation, that can be solved pointwise by a thresholding scheme (see [7]) ;
2. solving for $\mathbf{u}$ leads to a TV-L_2 minimization, also known as the Rudin-Osher-Fatemi (ROF) model, and for which exist fast algorithms (see for example [8]).

Because both subproblems can be solved pointwise and involve parallel computations, the authors of [7] successfully implemented their algorithm on Graphical Processing Units (GPU) for a significant speed-up.

1.1. Main contributions

Since optical flow is an important primitive for various Image Processing applications, such as shape-from-motion or video compression, there is a constant need for faster algorithms. In this work, we propose to solve the TV-L_1 optical flow of Eq. (2) directly, without any splitting but after smoothing its expression, to reduce the overall complexity. To do so, we base our approach upon a recent family of fast iterative numerical schemes for convex (eventually non-smooth) function minimization called Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [9].

Among the first order schemes (i.e. using only first order derivatives), this resolution scheme yields to iterations with an optimal convergence rate of $\mathcal{O}(1/n^2)$, which is one order of magnitude faster than Chambolle's method. Furthermore, we implemented the proposed algorithm on GPU using the industry standard OpenCL API to achieve realtime performance on a standard laptop hardware.

Note also that several recent papers presented optical flows designed for higher accuracy [10, 11, 12]. However, they involve some significant computational overhead (structure-texture decomposition, median filtering, bilateral or nonlocal filtering…) that are not harmless with respect to the speed of the computations, while we aim here at a faster algorithm.

2. FAST APPROXIMATED TV-L_1 OPTICAL FLOW

In this section, we present the proposed algorithm for fast TV-L_1 optical flow. To obtain faster convergence, we want to apply the iterations described in [9]. This requires to meet three conditions:

1. the function to minimize should be the sum of two convex functions, which is the case ;
2. at least one function should be Lipschitz-differentiable ;
3. one should known the solution of the minimization problem with respect to the other function.

First, we linearize the registration error of Eq. (2).Then, we introduce Nesterov's smooth approximation of a function [13] to the TV norm to make it Lipschitz-differentiable. Finally, we embed these two procedures into a FISTA-like procedure and solve the remaining minimization problem by applying a result from [7].

2.1. Problem linearization

In the sequel, we write $F(\mathbf{u})$ the function to minimize in Eq. (2):

$$F(\mathbf{u}) = \lambda\rho(\mathbf{u}) + \kappa(\mathbf{u}), \tag{4}$$

omitting the sum symbols for brevity, and with $\rho(\mathbf{u})$ and $\kappa(\mathbf{u})$ defined for each pixel location x as:

$$\begin{cases} \rho(\mathbf{u}) & = |I_0(x) - I_1\left(x + \mathbf{u}(x)\right)| \\ \kappa(\mathbf{u}) & = \|\mathbf{u}(x)\|_{\text{TV}} \end{cases}. \tag{5}$$

While $F(\mathbf{u})$ is convex, it is a non-linear function. Assuming that we want to measure small displacements from a previous estimate $\mathbf{u}_0$, we can take a Taylor expansion of $\rho(\mathbf{u})$ to linearize it:

$$\begin{aligned} \rho(\mathbf{u}) &= |I_0(x) - I_1\left(x + \mathbf{u}(x)\right)| \\ &\approx |I_0(x) - I_1(x + \mathbf{u}_0(x)) - \langle \nabla I_1, \mathbf{u} - \mathbf{u}_0 \rangle|. \end{aligned} \tag{6}$$

2.2. Smoothed Total Variation

Since the TV norm is non-smooth, we need to replace it by a smooth approximation. For each component u_d of the optical flow, we could add a small constant ϵ to the norm to avoid the singularity near 0, namely choosing $\|u_d\|_{\text{TV}} = \sqrt{\|\nabla u_d\|^2 + \epsilon^2}$.

However, we chose instead to follow the smoothing scheme proposed by Nesterov [13], because it comes with a value for the corresponding Lipschitz constant and does not induce any computational overhead. In this particular case it is similar to choosing a threshold μ on the TV norm that will make it switch from a quadratic behavior (for small values below μ) to the usual norm (values above μ).

Recall that the TV norm can also be defined in a *dual* form:

$$\kappa(u_d) = \|u_d\|_{\text{TV}} = \sqrt{\|\nabla u_d\|^2} = \max_{z \in Q_2} \langle z, D u_d \rangle, \tag{7}$$

where Q_2 is the unit ball in $\mathbb{R}^2$ and $D = (D_1, D_2)^T$ is the horizontal and vertical forward differences operator. We then introduce the smoothing parameter μ to balance between the quadratic and original behaviors to define the smoothed function κ_μ:

$$\kappa_\mu(u_d) = \max_{z \in Q_2} \langle z, D u_d \rangle - \frac{\mu}{2}\|z\|^2. \tag{8}$$

Eq. (8) admits a unique solution z_μ:

- if $\|\nabla u_d\|$is greater than μ, the first term dominates the expression and is maximum when the vectors u_d and z are aligned. Hence, in this case $z_\mu(u_d) = \nabla u_d / \|\nabla u_d\|$;
- if $\|\nabla u_d\|$ is smaller than μ, then the direction is again given by ∇u_d and setting the derivative of Eq. (8) to 0 yields $z_\mu(u_d) = \mu^{-1}\nabla u_d$.

This function is differentiable, and its gradient can be computed using the adjoint operator D^* (see [13] for the details):

$$\nabla \kappa_\mu(u_d) = D^* z_\mu(u_d). \tag{9}$$

It is immediate to see from Eq. (9) and the expression of z_μ that κ_μ is indeed Lipschitz-differentiable, and its Lipschitz constant L_μ is equal to the Lipschitz constant of D divided by μ: $L_\mu = 8/\mu$.

Note that since D was the *forward* differences gradient operator, D^* is *minus the backwards* divergence operator.

2.3. FISTA-based resolution scheme

Using the previous results, the objective function $F(\mathbf{u})$ now writes:

$$F(\mathbf{u}) = \lambda\rho_{\mathbf{u}_0}(\mathbf{u}) + \kappa_\mu(\mathbf{u}), \tag{10}$$

where $\rho_{\mathbf{u}_0}(\mathbf{u})$ stands for the linearized version of $\rho(\mathbf{u})$ from Eq. (6).

While standard first order iterative methods consider only the current estimate, the authors of [9] suggest to use also the previous iteration result to to achieve a faster convergence rate. The Iterative Soft Thresholding Algorithm (ISTA) is then applied to a mixture between the current and previous estimates, hence the name of Faster ISTA (FISTA).

Since κ_μ is Lipschitz-differentiable, we can apply the different steps of FISTA to $F(\mathbf{u})$. We call $\mathbf{y}$ the auxiliary variable and t the mixture parameter, and denote iteration by superscripts. The FISTA optical flow writes:

1. initialization: set $\mathbf{y}^1 = \mathbf{u}_0$, $t^1 = 1$;
2. for each iteration:
   (a) update the flow estimate from $\mathbf{y}$ by computing the result of the *proximal* operator of $\rho_{\mathbf{u}_0}$:

$$\mathbf{u}^k = P_{L_\mu}\{\rho_{\mathbf{u}_0}\}(\mathbf{y}) \tag{11}$$

   (b) update the mixture parameter:

$$t^{k+1} = \frac{1 + \sqrt{1 + 4(t^k)^2}}{2}; \tag{12}$$

   (c) update the auxiliary variable $\mathbf{y}$:

$$\mathbf{y}^{k+1} = \mathbf{u}^k + \frac{t^k - 1}{t^{k+1}}(\mathbf{u}^k - \mathbf{u}^{k-1}). \tag{13}$$

Writing $\mathbf{v}(\mathbf{y}) = \mathbf{y} - \frac{1}{L_\mu}\nabla\kappa_\mu(\mathbf{y})$, the proximal operator P_{L_μ} is defined here as the solution of the following minimization problem:

$$P_{L_\mu}\{\rho_{\mathbf{u}_0}\}(\mathbf{y}) = \operatorname*{argmin}_{\mathbf{u}} \left\{ \lambda\rho_{\mathbf{u}_0}(\mathbf{u}) + \frac{L_\mu}{2}\|\mathbf{u} - \mathbf{v}(\mathbf{y})\|^2 \right\}, \tag{14}$$

that can be solved by applying Prop. (3) from [7], leading to a simple thresholding operation with respect to $\tau = (\lambda/L_\mu)\|\nabla I_1\|^2$:

$$P_{L_\mu}\{\rho_{\mathbf{u}_0}\}(\mathbf{y}) = \mathbf{v}(\mathbf{y}) + \begin{cases} (\lambda/L_\mu)\nabla I_1 & \text{if } \rho_{\mathbf{u}_0}(\mathbf{v}) > \tau \\ -(\lambda/L_\mu)\nabla I_1 & \text{if } \rho_{\mathbf{u}_0}(\mathbf{v}) < -\tau \\ -\rho_{\mathbf{u}_0}(\mathbf{v})\nabla I_1 / |\nabla I_1| & \text{otherwise.} \end{cases} \tag{15}$$

3. EXPERIMENTS

3.1. Implementation details

We implemented the proposed algorithm in C++ using the OpenCV library [1], and on the GPU using the recently accepted OpenCL standard[2] on different laptops running the latest version of Mac OS X. While OpenCL implementations may not be as performant as other frameworks such as NVIDIA CUDA, it is compatible with a broader range of hardware, including mobile devices, and hence we hope for a wider compatibility in a near future.

Due to the linearization of Eq. (6), the scheme is valid for small flow vectors. Classically, we implemented a coarse-to-fine version of the algorithm, using the flow at the previous level as the linearization point $\mathbf{u}_0$.

For the GPU version, all the operations (downsampling, warping...) take place on the graphics unit. Since all the operations are pointwise, the implementation is straightforward. For better performance, we used the device embedded shared memory to avoid slow redundant memory accesses. The source code for the GPU version can be found on the authors website[3].

3.2. Complexity and running times

All the operations involved in the proposed algorithm can be computed pixel-wise. Furthermore, while the smooth TV gradient of Eq. (9) may seem complex, it amounts at computing the divergence of an image and thresholding it, which is also required when using for example Chambolle's projectors [8]. However, there is no loop additional loop than the iterations of FISTA at each scale, while Chambolle's method is iterative. Hence, few operations per pixel are involved at each iteration.

The GPU version is slightly optimized to limit the data exchanges between the host CPU and the GPU. We found the computation times to be very decent: on a standard laptop with the latest drivers and equipped with a mobile NVIDA GT320M device, we were able to process 8 frames per second from the built-in webcam, using 3 scales and 100 iterations per scale. The speed of our demonstration software (see Section 4) was actually limited by the user interface updates.

3.3. Optical flow results

Figure 1 shows a comparison between the proposed scheme and an exact TV-L_1 algorithm (we took [7] as reference). While the asymptotic error is very similar, one can see the benefit of using FISTA in the fast error decrease. Note also in Fig 1 that the approximation for small values of μ is very good, while the solver does converge one order of magnitude faster than standard iterations.

4. REALTIME VISUAL INTERACTION

4.1. Example application

To demonstrate the suitability of our algorithm for realtime applications, we use it to track user gestures in a camera-based interactive setup. As shown in Fig. 3, the user can move an image according to the mean optical flow computed inside its frame (materialized by the green square). When the picture reaches the target area (white square), it is zoomed and displayed full screen. This basic setup allows us to assess the behavior of the optical flow algorithm and tune its parameters. Since we do not need highly accurate results, we set the algorithm to take only one linearization point per scale, and run 100 iterations for each of the 3 levels.

[1] http://opencv.willowgarage.com/
[2] http://www.khronos.org/opencl/
[3] http://lts2www.epfl.ch/people/dangelo/

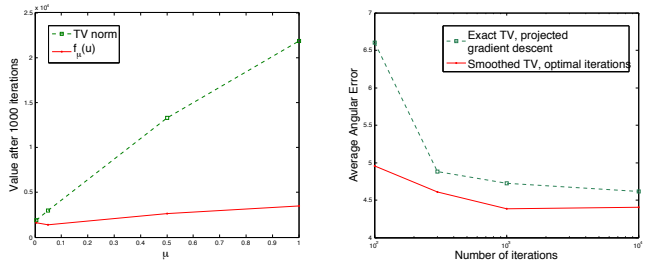

Fig. 1. *Left:* evaluation of the TV approximation. For small values of μ, our approximation is very close to the actual TV norm (dashed curve). *Right:* comparison with the baseline TV-L_1 algorithm in [7]. While the asymptotic error is the same, our solver does converge faster.

Fig. 2. Flow example for a frame from the Middlebury dataset [14].

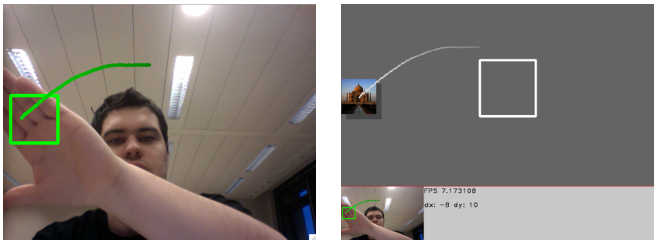

Fig. 3. Basic setup using only the optical flow to control one image.

For more complex interactions when several pictures are present, more gestures are available: for instance, swiping horizontally across the screen allows to navigate among the images. In this case, several objects and motions may interfere: since the two hands (and possibly some background objects) are moving at the same time, the TV constraint in the flow estimation may propagate and mix these motions together.

4.2. Improving the user experience

In this version, we solve this problem by adding skin and face detection to the application. These are executed on the CPU in parallel to the optical flow, which resides on the graphics unit. Skin detection

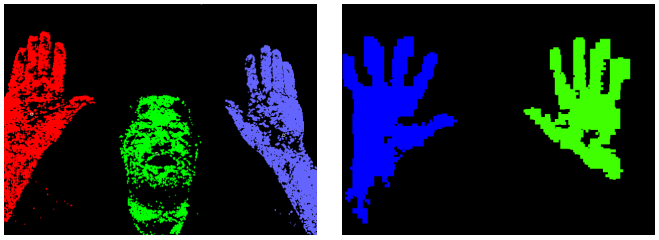

Fig. 4. Output of the skin detection and clustering procedure. *Left:* using back-projection and k-means on the original frame. *Right:* the k-means algorithm was used in a coarse-to-fine setup and the face was removed after detection.

is used to filter out all the non relevant areas from the input frames, whereas face detection allows us to keep only the hands of the user. The result of this process is then applied as a binary mask to to select regions of interests in the optical flow.

After an initialization step, the skin detector works by color histogram matching to classify the pixels as skin / non-skin, followed by k-means clustering to group the detections in three blobs (right hand, left hand and head). For the face detection part, we used the OpenCV implementation of the standard Viola - Jones detector [15].

Despite the numerous computations, we process the input video feed at around 8 frames-per-second on a recent laptop equipped with an NVIDIA GT320M unit, providing a smooth user experience. This clearly shows the interest of leveraging the GPU for computation-intensive tasks, while additional processing is performed by the host hardware.

Additionally, we found that it was not intuitive to immediately stop an object when the user ceases dragging it. Adding some inertia instead to produce deceleration trajectories improved the user experience.

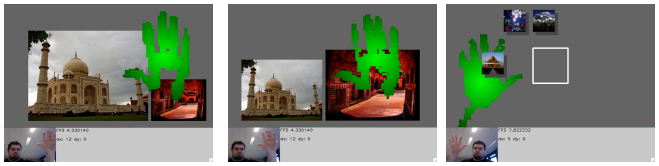

Fig. 5. Using our software. On the leftmost illustrations, the framerate is lowered by the scaling computation of the displayed images, not by the optical flow. An obvious improvement would be to port this part of the interface to the GPU too.

5. CONCLUSION AND FUTURE WORK

In this work, we have demonstrated the feasibility of a very fast, numerically optimal, TV-L_1 optical flow algorithm. While it shows similar precision to the reference TV-L_1 method, its convergence speed is one order of magnitude higher. Hence, it can be stopped after fewer iterations. Furthermore, it does not break the parallelism of modern TV-L_1 solvers and can be implemented on GPUs. Put together, these properties allowed us to achieve realtime performance and to embed this algorithm into a more complex software.

As a future work, we will work on implementing the "secrets" for more accurate optical flow recently proposed in [12]. While this should be straightforward for the structure-texture decomposition, which can also be implemented on GPU, there may be some performance issues with the median filtering. Finally, we also plan to extend the gesture recognition capacities of our demonstration software by adding it some finger position recognition capabilities.

6. REFERENCES

[1] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[2] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow," *Proceedings of the Fourth International Conference on Computer Vision*, pp. 231–236, 1993.

[3] E. Mémin and P. Pérez, "Dense estimation and object-based segmentation of the optical flow with robust techniques," *IEEE Transactions on Image Processing*, vol. 7, no. 5, pp. 703–719, 1998.

[4] R Deriche, P Kornprobst, and G Aubert, "Optical-flow estimation while preserving its discontinuities: a variational approach," in *Proceedings of the Second Asian Conference on Computer Vision*, 1995, pp. 290—295.

[5] I. Cohen and I. Herlin, "Non uniform multiresolution method for optical flow and phase portrait models: Environmental applications," *International Journal of Computer Vision*, vol. 33, no. 1, pp. 29–49, 1999.

[6] J. Weickert and C. Schnörr, "Variational optic flow computation with a spatio-temporal smoothness constraint," *Journal of Mathematical Imaging and Vision*, vol. 14, no. 3, pp. 245–255, 2001.

[7] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *Proceedings of the 29th DAGM conference on Pattern Recognition*, 2007, pp. 214–223.

[8] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1, pp. 89–97, 2004.

[9] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[10] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, "An improved algorithm for tv-l 1 optical flow," in *Statistical and Geometrical Approaches to Visual Motion Analysis*, pp. 23–45. Springer, 2009.

[11] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, "Anisotropic Huber-L1 optical flow," in *Proceedings of the British Machine Vision Conference*, 2009.

[12] D. Sun, S. Roth, and M.J. Black, "Secrets of optical flow estimation and their principles," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 2432–2439.

[13] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, 2005.

[14] S. Baker, D. Scharstein, JP Lewis, S. Roth, M.J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, November 2010, pp. 1–8.

[15] P Viola and M Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2002.