DECOUPLED COARSE-TO-FINE MATCHING AND NONLINEAR REGULARIZATION FOR EFFICIENT MOTION ESTIMATION

Mariano Tepper, Guillermo Sapiro

Department of Electrical and Computer Engineering, University of Minnesota, USA

ABSTRACT

A simple motion estimation algorithm, light-weighted both in memory and in time, is presented in this paper. This simplicity is achieved by decoupling the matching and the regularization stages in the estimation process. Experiments show that the obtained results are comparable with state-of-the-art algorithms that are much more computationally demanding.

***Index Terms*—** Motion estimation, dense SIFT, matching, PatchMatch, nonlinear filtering.

1. INTRODUCTION

Motion estimation is one of the most important low-level components in biological visual systems. In many cases, e.g., animals of prey, it even takes preference over stereo vision. Gestalt psychologists consider it as one of the primary laws, i.e., the law of common fate, that govern human vision.

In computer vision, the motion between two adjacent frames in a video is often represented as a 2D flow field, a.k.a. optical flow. Variational methods have become the prevalent technique to estimate dense optical flow in today's computer vision literature, e.g., see Brox and Malik [1].

The SIFT-Flow algorithm [2] computes the flow field between two dense SIFT (DSIFT) images, while preserving spatial discontinuities. SIFT is a local descriptor that characterizes local gradient information [3]. It was originally intended as a sparse representation (i.e., at keypoints), but has been widely used later as a dense (i.e., pixel-wise) image descriptor. When used in this context it is often referred to as Histogram of Oriented Gradients (HOG).

The authors of SIFT-Flow claim that their algorithm is not devised to align images that are densely sampled in time (by densely we mean at video rate), that is for motion estimation, but for images that are densely sampled in the space of all images [2]. This can be considered puzzling, since densely sampling in the space of all images *necessarily* means densely sampling in time. Nonetheless, SIFT-Flow is able to align images that exhibit motions beyond the very small displacement examples found in some standard optical flow datasets.

Formally, SIFT-Flow can be described as follows. Let $I : \Omega \to Y$ be an image. Without loss of generality, in this work we consider $\Omega \subseteq \mathbb{N} \times \mathbb{N}$ and $Y \subseteq \mathbb{R}$. Let $D : \Omega \to \mathbb{R}^{128}$ be the DSIFT image of image I. Let also ε be a spatial neighborhood relation in I. Let us denote by $\mathbf{w} : \Omega \to \Omega$ the flow field between two DSIFT images D_a and D_b (corresponding to two video frames I_a and I_b, respectively), being u, v the horizontal and vertical components of $\mathbf{w}$, respectively. SIFT-Flow seeks to minimize the following energy function

$$E(\mathbf{w}) = \sum_{\mathbf{p}\in\Omega} \min\Big(\|D_a(\mathbf{p}) - D_b(\mathbf{p}+\mathbf{w}(\mathbf{p}))\|_1, t\Big) + \quad (1)$$

$$\sum_{\mathbf{p}\in\Omega} \eta\Big(|u(\mathbf{p})| + |v(\mathbf{p})|\Big) + \quad (2)$$

$$\sum_{(\mathbf{p},\mathbf{q})\in\varepsilon} \min\Big(\alpha|u(\mathbf{p}) - u(\mathbf{q})|, d\Big) + \min\Big(\alpha|v(\mathbf{p}) - v(\mathbf{q})|, d\Big), \quad (3)$$

where $t, \eta, \alpha, d \in \mathbb{R}$ are parameters of the method. $E(\mathbf{w})$ contains (1) a data term, (2) a small displacement term and (3) a smoothness term.

SIFT-Flow does not provide the level of subpixel accuracy of modern optical flow methods (e.g., LDOF [1]). However, in many applications, achieving high precision is not necessary but speed is.

Our goal in this work is to develop a light-weight method that can be implemented in real time and that provides accuracy levels that are on par with SIFT-Flow. We show that this can be done by decoupling the data term and regularity terms. We thus characterize the problem as a matching step plus a smoothing step. In §2 and §3 we cover the former and the latter respectively. In §4 we present experimental validation, and then draw some conclusions in §5.

2. MOTION ESTIMATION AS A REGULARIZED NEAREST-NEIGHBORHOOD PROBLEM

PatchMatch [5] is a method to perform an approximate Nearest Neighbor (NN) search between all the patches in two given images. It is composed two steps that are run iteratively: a random search step and a propagation step. It was shown to provide both good performance and efficiency. To the best of our knowledge, Boltz and Nielsen first proposed to use PatchMatch for motion estimation [6]. Following SIFT-Flow, in-

Work partially supported by NSF, ONR, NGA, ARO, and NSSEFF.

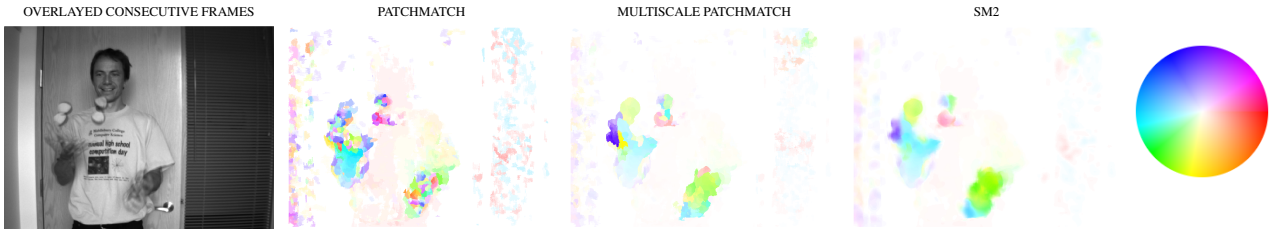

Fig. 1: Each step of the proposed algorithm helps in regularizing the estimated motion with respect to the output of PatchMatch. For visualization, we represent each flow vector with a color pixel whose hue and saturation correspond to the orientation and magnitude of the vector, respectively [7].

stead of simply comparing raw image patches, we propose to compare SIFT features computed at each pixel. Thus, we first minimize the following energy function

$$E(\mathbf{w}) = \sum_{\mathbf{p}\in\Omega} \|D_a(\mathbf{p}) - D_b(\mathbf{p}+\mathbf{w}(\mathbf{p}))\|_1. \qquad (4)$$

In its random step, PatchMatch tries to replace the current NN candidate by other randomly chosen candidates. We have observed that constraining this step as follows provides smoother results: the random NN candidate must improve the current NN by more than 20% to replace it. This plays the role, in some sense, of the small displacement and smoothness terms in SIFT-Flow, at a lower computational cost.

To address the patch size selection problem, Boltz and Nielsen propose a hierarchical segmentation tree that propagates good matches [6], instead of using spatial neighborhood in the image. They correctly state that the closer the segmentation is to a motion segmentation, the better the final result will be. However this is a chicken-and-egg problem and does not fully solve the size selection issue.

Multiscale approaches, in their coarse-to-fine flavor, have been successfully used for motion estimation [1, 2]. We therefore propose to build a Gaussian pyramid of the original video frames, then compute DSIFT images at each level, and finally compute PatchMatch in the coarsest level, recursively propagating the estimated NN field as an initialization for finer levels. This also serves in practice to capture large displacements with small search windows. In our experiments we found that in general using 3 levels was sufficient. Fig. 1 depicts an example of the regularity introduced by this multiscale process (compare center left and center right images).

Since there are very efficient $O(n)$ algorithms for exact DSIFT computation [8], where n is the number of pixels, and considering the number of levels in the pyramid as a small constant, the complexity of our algorithm is $O(kln)$, compared to SIFT-Flow's which is $O(kl^2n)$, where k is the number of iterations, and l is the window size. Furthermore, all the steps in our algorithm can be implemented in real time using GPGPU [5].

3. REGULARIZING THE FLOW

Although proven to be beneficial in every motion estimation algorithm, smoothness/regularization is a dangerous constraint. The amount of smoothness introduced in the estimation process is a critical parameter of most methods that, if incorrectly set, can completely destroy the accuracy of the output. This is due to the fact that motion is often regarded as a trade-off between fitting and smoothness.

We claim that smoothness is more important than fitting in certain regions of a scene, while the inverse situation occurs in other regions. These regions are spatially separated by sharp motion edges. This is why isotropic smoothness is doomed to fail and hindering its effects becomes such an art.

This is particularly true in DSIFT images. Let us first define the sparsity image D_0 of the DSIFT image D as $D_0(\mathbf{x}) = \|D(\mathbf{x})\|_0$ (this new feature will be exploited for smoothing below). As can be observed in Fig. 2, edge and non-edge SIFT features belong to two different D_0 distributions. This is also related to the first uses of SIFT as features at sparse locations [3]: SIFT are informative near edges or texture.

Since smoothness should be encouraged between certain points and avoided between others, and following Xiao et al. [4], we turn our attention to nonlinear smoothing techniques. We are particularly interested in the bilateral filter (BF) [9] due to its performance and because efficient $O(1)$ implementations exist, see Chaudhury et al. [10] and references therein. Let $f : \Omega \to Y$, with support S, and $g : Y \to Y$ be two smoothing kernels (f is a spatial kernel and g a range kernel). The BF is defined as

$$\tilde{I}(\mathbf{p}) = \kappa^{-1} \sum_{\mathbf{q}\in S} f(\mathbf{q})\, g\Big(I(\mathbf{p}) - I(\mathbf{p}+\mathbf{q})\Big)\, I(\mathbf{p}+\mathbf{q}), \quad (5)$$

where $\kappa = \sum f(\mathbf{q})\; g\left(I(\mathbf{p}) - I(\mathbf{p}+\mathbf{q})\right)$ is a normalizing term. Let $I' : \Omega \to Y'$ be another image with the same domain but a different range than I. By replacing the range kernel g by a different smoothing function $h : Y' \to Y$ applied to I', we get the cross BF, whose range locality is determined by I'. In our case we set $I' = D_0$ and apply the filter to the horizontal and vertical components of the motion field, u and v, regularizing them using the sparsity of the DSIFT image as an *anisotropic* constraint:

$$\tilde{u}(\mathbf{p}) = \kappa_u^{-1} \sum_{\mathbf{q}\in S} f(\mathbf{q})\, h\Big(D_0(\mathbf{p}) - D_0(\mathbf{p}+\mathbf{q})\Big)\, u(\mathbf{p}+\mathbf{q}), \quad (6)$$

where κ_u^{-1} is modified accordingly. We omit the equation for v as it is the very same as for u.

	Dimetrodon		Grove2		Grove3		Hydrangea		RubberWhale		Urban2		Urban3		Venus	
	SIFT-F	SM2	SIFT-F	SM2	SIFT-F	SM2	SIFT-F	SM2	SIFT-F	SM2	SIFT-F	SM2	SIFT-F	SM2	SIFT-F	SM2
mean EE	0.420	0.427	0.504	0.510	0.971	1.007	0.560	0.563	0.357	0.357	0.820	0.899	1.106	1.395	0.465	0.508
var EE	0.059	0.054	0.292	0.202	2.305	2.019	1.273	1.182	0.185	0.162	2.630	3.269	4.413	5.661	0.824	0.722
med EE	0.399	0.403	0.411	0.419	0.473	0.501	0.194	0.197	0.245	0.246	0.472	0.490	0.453	0.519	0.250	0.250

Table 1: Endpoint Error (EE) statistics for scenes in the Middlebury dataset with available ground truth. SIFT-F stands for SIFT-Flow.

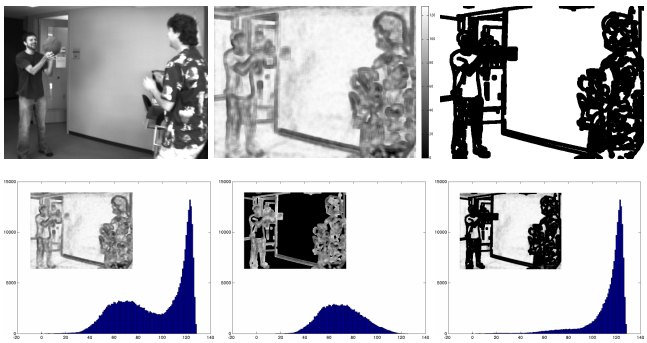

Fig. 2: Top, from left to right: original image, DSIFT sparsity, and dilated Canny edges used as a mask to separate edges from non-edges. Bottom, from left to right: DSIFT sparsity histogram of the whole image, restricted to edges, and restricted to non-edges.

In this way we enforce that motion is averaged between pixels that share a "common structure," while preventing it otherwise. We filter the flow field obtained at each level of the multiscale algorithm before using it in the next level. Fig. 1 depicts an example of the regularity obtained with this filtering technique (compare center right and right images).

We call the proposed algorithm Smoothed Multiscale SIFT Matching (SM2). The BF can be implemented in $O(n)$ [10], thus the overall time complexity of SM2 is $O(kln)$ (recall that PatchMatch usually converges in about $k = 5$ iterations [5]). The space complexity of SM2 is also low because of its components' low memory requirements.

4. EXPERIMENTAL VALIDATION

All SIFT-Flow results were computed with the MATLAB package released by its authors (the core functions are mex files).[1] We use a pure MATLAB implementation of the BF.[2] For computing DSIFT we tried the implementation included in the SIFT-Flow package but got slightly better results using VLFeat [11] (note that the former computes features at the image borders while the latter does not.). We implemented PatchMatch as a mex file, based on Barnes et al.'s c++ minimal code.[3] All experiments were run on a MacBook Pro with a 2.7GHz Intel Core i7 processor.

The bin size in DSIFT was set to 3 pixels. For all experiments we retained SIFT-Flow's default parameters, except the search window size and the number of levels. We tested our algorithm on the Middlebury [7] and moseg [12] datasets. For the Middlebury dataset, we set these parameters to 4 pixels and 3 levels, respectively. These same values were used for testing SM2. When testing with the moseg dataset, SIFT-Flow was unable to provide satisfactory results with a search window of 4 pixels, while changing it to 10 pixels provided good results. For SM2 we saw no need to change this value, and 4 pixels were used.

Fig. 3 depicts a few examples of the obtained results. In most cases the flows obtained with both algorithms are very similar, with small differences on the motion boundaries, as can be observed on the last column. As expected, SIFT-Flow produces smoother flow fields. However it is unable to detect some fast motions (e.g., the feet on the second and fifth rows). The milder regularity constraints in SM2 allow to detect them. Notice that on the fourth row, SIFT-Flow hallucinates a separate motion on the trees where SM2 does not. The Endpoint Error [7] statistics, presented in Tab. 1, show that both methods have similar accuracy.

On the Middlebury dataset SIFT-Flow runs in 25.7 s and SM2 in 13.3 s per pair of frames. On the moseg dataset (for long sequences, we used only the first 50 frames), SIFT-Flow runs in 21.4 s and SM2 in 10 s per pair of frames. In these experiments, and without fine tuning of our implementation, SM2 is approximatively twice as fast as SIFT-Flow. Of SM2's running time, approximatively 23% and 71% are spent on computing DSIFT and PatchMatch, respectively. By using a simplified SIFT, with 2×2 bins and 4 orientations, we can further improve the speed fivefold while decreasing accuracy by less than 5% on the Middlebury dataset.

5. CONCLUSIONS

From the conceptual point of view, we have shown that it is possible to decouple the matching and the regularization terms in a motion estimation algorithm without significant loss in precision. We believe that this opens new leads for the development of more efficient computer vision algorithms.

Based on this idea, we have presented a new algorithm, named SM2, for estimating the motion between two images. Its accuracy is on par with SIFT-Flow while being much faster. The memory footprint of SM2, contrarily to SIFT-Flow, is small. SM2 is also capable of capturing fast motions because of its milder and adaptive regularity constraints.

As future work, we plan to implement SM2 in real time using modern BF algorithms and GPGPU. SM2 suffers from a widespread problem in motion estimation algorithms: large occlusions are is not handled correctly. We plan on investigating this cue to further refine our results.

[1] http://people.csail.mit.edu/celiu/SIFTflow/
[2] http://people.csail.mit.edu/jiawen/
[3] http://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/

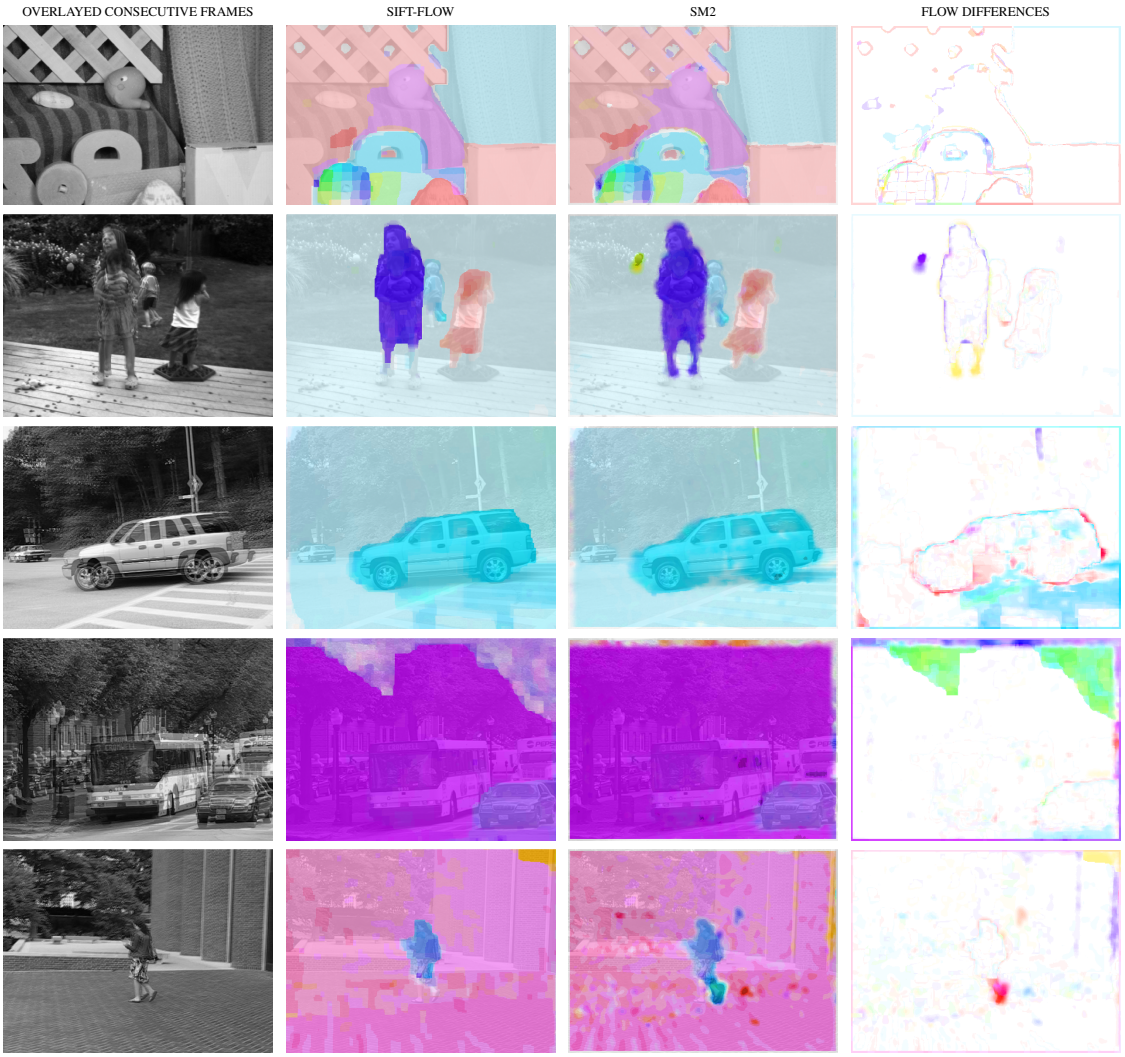

Fig. 3: Comparison between SIFT-Flow and the proposed SM2. In general, both methods produce similar results, being SIFT-Flow's smoother. In some cases, the true motion is underestimated by SIFT-Flow, while being more accurately recovered by SM2. The color code is explained in Fig. 1.

6. REFERENCES

[1] T. Brox and J. Malik, "Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation," *IEEE TPAMI*, vol. 33, no. 3, pp. 500–513, Mar. 2011.

[2] C. Liu, J. Yuen, and A. Torralba, "SIFT Flow: Dense Correspondence across Scenes and Its Applications," *IEEE TPAMI*, vol. 33, no. 5, pp. 978–994, May 2011.

[3] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[4] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," in *ECCV*, May 2006, pp. 211–224.

[5] C. Barnes, D. Goldman, E. Shechtman, and A. Finkelstein, "The PatchMatch randomized matching algorithm for image manipulation," *Comm ACM*, vol. 54, pp. 103–110, Nov. 2011.

[6] S. Boltz and F. Nielsen, "Randomized motion estimation," in *ICIP*, 2010, pp. 781–784.

[7] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. Black, and R. Szeliski, "A Database and Evaluation Methodology for Optical Flow," *IJCV*, vol. 92, pp. 1–31, Mar. 2011.

[8] M. Grabner, H. Grabner, and H. Bischof, "Fast approximated SIFT," in *ACCV*, Jan. 2006, pp. 918–927.

[9] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV*, Jan. 1998, pp. 839–846.

[10] K. N. Chaudhury, D. Sage, and M. Unser, "Fast O(1) Bilateral Filtering Using Trigonometric Range Kernels," *IEEE TIP*, vol. 20, no. 12, pp. 3376–3382, Dec. 2011.

[11] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[12] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *ECCV*, Sept. 2010, pp. 282–295.