Scale-Space SIFT Flow

Weichao Qiu[1,2], Xinggang Wang[1,2], Xiang Bai[1], Alan Yuille[2], and Zhuowen Tu[3]
[1]Dept. of Electronics and Information Engineering, Huazhong Univ. of Science and Technology
[2]Department of Statistics, UCLA
[3]Department of Cognitive Science, UCSD
qiuwch@gmail.com, {xgwang, xbai@hust.edu.cn}, yuille@stat.ucla.edu, ztu@ucsd.edu

Abstract

The state-of-the-art SIFT flow has been widely adopted for the general image matching task, especially in dealing with image pairs from similar scenes but with different object configurations. However, the way in which the dense SIFT features are computed at a fixed scale in the SIFT flow method limits its capability of dealing with scenes of large scale changes. In this paper, we propose a simple, intuitive, and very effective approach, Scale-Space SIFT flow, to deal with the large scale differences in different image locations. We introduce a scale field to the SIFT flow function to automatically explore the scale deformations. Our approach achieves similar performance as the SIFT flow method on general natural scenes but obtains significant improvement on the images with large scale differences. Compared with a recent method that addresses the similar problem, our approach shows its clear advantage being more effective, and significantly less demanding in memory and time requirement.

1. Introduction

Methods that are based on dense features have significantly advanced the field of image matching and have been widely adopted in various computer vision applications, *e.g.*, scene parsing [12, 20], image/video retrieval [13], motion estimation [13], and depth estimation [21]. Compared to the previous approaches using extracted sparse features, e.g. interest points [17, 2], dense features are better to preserve the intrinsic uncertainties so that more robust decisions can be made in the later stages; in practice, significant improvements over the previous method in image matching and classification have been widely observed by SIFT flow [13] and Spatial Pyramid Matching [8] respectively.

Sparse features, *e.g.* the SIFT points [16] or Harris corners [17], can somewhat automatically (to a certain degree) determine the feature scale. They work well for matching identical objects in the images but fail to deliver reliable results for semantic image alignment in the more general cases. In matching, it is hard to determine the right scales unless both the source and target images are observed.

When applying the dense features, the existing methods [13, 16] mostly compute the features at a fixed scale. In this paper, we study the image matching/correspondence/registration problem and focus on the SIFT flow algorithm [13]. One popular method for computing the dense features is the Dense SIFT (DSIFT) descriptor which extracts SIFT histogram at a single scale for all pixels with overlapping patches. Using DSIFT, SIFT flow [13] aligns two images by minimizing matching cost and keeping the flow field smooth. Since the DSIFT feature is only computed at one scale in SIFT flow, it requires that objects in two images share the same/similar scales. This makes SIFT flow problematic in dealing with images of large scale change, which was observed in [6]. To overcome this problem, a recent method called Scale-Less SIFT (SLS) was proposed in [6]. When performing the matching, SLS extracts a set of DSIFT features at multiple scales for every pixel and uses set-to-set distance to measure the matching cost between the corresponding pixels. More specifically, SLS uses Projection Frobenius Norm to compute the set-to-set distance. In practice, if there are $100,000$ pixels in each image then a locality constraint is needed; running SLS with the Projection Frobenius Norm is both time- and memory-consuming. For an image of standard size 640×480, it consumes more than $10G$ memory and takes hours to perform one matching, which makes the SLS algorithm nearly impractical to scale up. In addition, SLS tries to address the scale difference problem by feature fusion. Instead, we tackle this problem by creating a scale field to automatically explore the best SIFT matches at the right scale at each location. This simple but intuitive solution yields significant improvement over SLS in performance, speed, and memory consumption; it also demonstrates its advantage over the original SIFT flow method when dealing with images of large scale differences.

The "Scale-space" theory was firstly introduced by Witkin in [23] for multi-scale image representation and now has been considered as a well-studied topic [24, 9]. Linear Gaussian scale-space is the most widely used multi-scale image representation method. Anisotropic diffusion [18]

was proposed to obtain non-linear smooth scale-space by Perona and Malik. Nevertheless, it is beneficial to look back at some of the early work in computer vision whose essences are often inspiring to the modern vision algorithms.

In our approach, we associate each pixel with a scale factor; scale factors on the image lattice form a scale field; dense feature for each pixel is estimated for a particular scale during matching. The "best" match is estimated with the selection of the right scale factors through (1) minimizing feature matching cost, (2) keeping the flow field smooth, and (3) keeping the scale field smooth. Fig. 1 shows the clear advantage of introducing the scale field of our method which naturally explores the scale space for matching image pairs of different scales. We give an iterative solution for minimizing the objective function.

In addition, we will show in the experiments that our method does not degrade when dealing with image scenes of small scale changes. We also apply our method to example-based color enhancement.

2. Related Work

The related work can be roughly divided into categories on scale invariant image representations and and those performing image matching/registration. To build scale invariant image representations, most methods are based on automatic scale selection which seeks for each feature point a stable and characteristic scale. Automatic scale selection methods usually search for local extrema in the 3D scale-space representation of an image (x, y and scale). This idea was introduced in the early eighties by Crowley [3]. Based on this idea, Laplacian of Gaussian (LoG) [10] and Difference of Gaussian (DoG) [16] are widely used. Please see [17] for a more comprehensive survey for automatic scale selection. In [7], a scale invariant image descriptor is build without scale selection. Scale invariance of SID is guaranteed in the Fourier Transform Modulus. There are a large number of image matching/aligment/registration methods in computer vision and medical imaging *etc*. Some recent works include: the dense graph matching method in [15], the vector field learning method in [25], factorized graph matching [26] *etc*. Our method is built upon the SIFT flow [14, 13] method, which does not consider the scale change problem.

3. Approach

3.1. SIFT Flow Review

We first review the SIFT flow formulation in [13]. In SIFT flow, it assumes image matching are performed without much scale change. Let $\mathbf{p} = (x, y)$ be the grid coordinate of images, $\mathbf{w}(\mathbf{p}) = (u(\mathbf{p}), v(\mathbf{p}))$ be the flow vector at $\mathbf{p}$, and s_1 and s_2 be two SIFT images that we want to match. $s_1(\mathbf{p})$ denotes a SIFT descriptor at position $\mathbf{p}$ of SIFT image s_1. Set ϵ contains all spatial neighborhoods (usually a four-neighbor system is used). The energy function for SIFT flow is defined as:

$$\begin{aligned} E(\mathbf{w}) = &\sum_{\mathbf{p}} \min\left(\| s_1(\mathbf{p}) - s_2(\mathbf{p}+\mathbf{w}(\mathbf{p})) \|_1, t\right) + \\ &\sum_{\mathbf{p}} \eta\left(|u(\mathbf{p})| + |v(\mathbf{p})|\right) + \\ &\sum_{(\mathbf{p},\mathbf{q})\in\epsilon} \min(\alpha|\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) \end{aligned} \tag{1}$$

which contains a *data term*, a *small displacement term* and a *smoothness term* (*a.k.a* spatial regularization). In Eqn. (1), $\| s_1(\mathbf{p}) - s_2(\mathbf{p}+\mathbf{w}(\mathbf{p})) \|_1$ is the *data term* which constrains the SIFT descriptors to be matched along with the flow vector $\mathbf{w}(p)$. $|u(\mathbf{p})| + |v(\mathbf{p})|$ is the *small displacement term* which constrains the flow vectors to be as small as possible. $|\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})| = |u(\mathbf{p}) - u(\mathbf{q})| + |v(\mathbf{p}) - v(\mathbf{q})|$ is the *displacement smoothness term* which constrains the flow vectors of adjacent pixels to be similar. In this objective function, truncated ℓ_1 norms are used in both the data term and the displacement smoothness term to account for matching outliers and flow discontinues, with t and d as threshold, respectively.

In [13], the optimization problem in Eqn. (1) was solved using a dual layer loopy belief propagation; a coarse-to-fine matching scheme is further adapted which can both speed up matching and obtain a better solution. There is no scale factor in Eqn. (1), while in many dense feature matching applications, images are at different scales. In SIFT flow, dense SIFT feature computed in fixed grids and fixed patch size in image can handle small scale variation. From our experimental experience, SIFT flow can handle scale variation with the ratio in $[1/1.5, 1.5]$. In the following section, we proposed a dense feature matching formulation by simply associating each pixel in image with a scale factor.

3.2. Scale-Space SIFT Flow

When matching two images, we keep the second image at its own scale. In the first image, we assign a scale factor for every position which is denoted by $\sigma(\mathbf{p}) \in \mathbf{R}$. We denote $s_1(\mathbf{p}, \sigma(\mathbf{p}))$ as a SIFT feature computed at scale $\sigma(\mathbf{p})$ at position $\mathbf{p}$ in the first image. For simplicity, we omit the scale factor if it is 1. Our new energy function is given as follows:

$$\begin{aligned} E(\mathbf{w}, \sigma) = &\sum_{\mathbf{p}} \min\left(||s_1(\mathbf{p}, \sigma(\mathbf{p})) - s_2(\mathbf{p}+\mathbf{w}(\mathbf{p}))||_1, t\right) + \\ &\sum_{\mathbf{p},\mathbf{q}\in\epsilon} \min(\alpha|\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) + \\ &\sum_{\mathbf{p},\mathbf{q}\in\epsilon} \min(\beta|\sigma(\mathbf{p}) - \sigma(\mathbf{q})|, \tau) \end{aligned} \tag{2}$$

(a) Source image (b) Target image (c) SIFT flow (d) Our method (e) Estimated scale field

Figure 1. We find dense correspondence from (a) to (b), then use dense correspondence to warp the source image to the target image. (c)(d) are warped image from SIFT flow and our method respectively. (e) shows the estimated scale field.

In Eqn. (2), besides a *flow field* $\mathbf{w}$, we have another *scale field* σ. We require both fields to be smooth. In Eqn. (2), $||\sigma(\mathbf{p}) - \sigma(\mathbf{q})||$ is the *scale smoothness term* which constrains the scale factors of adjacent pixels to be similar. Different from the energy function in SIFT flow, here we remove the small displacement term. This is due to the fact that the displacements are no longer small when objects of interest to be matched are at different scales. Another advantage of removing the small displacement term is that it allows object of interest appears in any position of the second image. This gives more flexibility than before. Parameters β and τ are used for adjusting the weight of scale smoothness term and truncating scale smoothness term. To deal with potential outliers in the scale field, we use ℓ_1 norm for scale smoothness term.

Our formulation gives a very natural and intrinsic approach to deal with the scale variation issue in dense feature matching. Rather than computing multi-scale DSIFT descriptors and matching them using set-to-set distance in [6], we aim at computing feature at a proper scale. We allow different positions have different scales, and constrains the final scale field is smooth. This setting is very reasonable.

3.3. Optimization for (2)

In this subsection, we give our optimization method for our energy function in (2). A straightforward solution is to put the scale factor $\sigma(\mathbf{p})$ in the dual layers loopy belief propagation framework. However, because of the coarse-to-fine scheme needs to re-scale the images, it is not suitable for optimizing the scale factor. Alternatively, we propose an iterative solution. When the scale field σ is fixed, we optimize the flow field $\mathbf{w}$; when the flow field $\mathbf{w}$ is obtained, we optimize the scale field σ. To tackle the problem of optimizing scale field, we discrete the scale space into N states, denoted as $\zeta = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$. The estimated scale factor for every position should be in the scale space, denoted as $\sigma(\mathbf{p}) \in \zeta$. To clearly illustrate our algorithm, we also define an index map $\mathbf{m} = \{m(\mathbf{p}) \in [1, \ldots, N]\}$ for the scale field, thus $\sigma(\mathbf{p}) = \sigma_{m(\mathbf{p})}$.

The whole solution is outlined in Fig 2. For image 2, we always compute SIFT image at its original scale. In the initialization stage, for every scale in the scale space, we compute SIFT image for the first image at this scale, match the SIFT image to the SIFT image 2 using SIFT flow formulated in Eqn. (3), and obtain the data term for every position at current scale. Then we initialize scale factor for every position by looking for smaller data term and keeping scale smooth at the same time; these two objectives are formulated together in a MRF framework given in Eqn. (4). After the scale field is initialized, we iteratively optimize the flow field and the scale field. Now, the idea of optimizing flow field is fixing SIFT feature as $s_1(\mathbf{p}, \sigma_{m(\hat{\mathbf{p}})})$ in the first image and computing SIFT flow; this is formulated in Eqn. (5). The idea of optimizing scale field is fixing flow as $\hat{\mathbf{w}}(\mathbf{p})$ and looking for smaller data term across scales and keeping the scale space smooth at the same time; this is formulated in Eqn. (6).

The problems of optimizing flow field (in Eqn. (3) and Eqn. (5)) and optimizing scale field (in Eqn. (4) and Eqn. (6)) are easy to tackle using existing optimizing technics. We directly use optimization method in SIFT flow [13] to optimize flow field. We use the efficient belief propagation algorithm in [4]; coarse-to-fine scheme is not used when optimizing the scale field, since the scale space is very small compared to the displacement space.

4. Experiment

In our experiments, we report the image matching results by our method on benchmark datasets and compare with SIFT flow [13] and SLS [6], which are most related. For a quantitative evaluation, we use the standard Middlebury dataset [1], which has been widely used as benchmark for optical flow algorithms. Then we quantitatively and qualitatively compare the matching performance to [6] on the identical images used in [6]. Furthermore, we apply our algorithm to example-based color enhancement.

Through our experiment, the scale space of the dense SIFT is set to $\zeta = \{1, 2, 4, 6, 8\}$; the weight parameters $\alpha = 3$, $\beta = 60$; the threshold for flow field $d = 60$; for

Initialization: For $n \in [1, \ldots, N]$, we compute flow field by solving:

$$\hat{\mathbf{w}}_n = \arg\min_{\mathbf{w}} \sum_{\mathbf{p}} \min\left(||s_1(\mathbf{p}, \sigma_n) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p}))||_1, t\right) + \sum_{\mathbf{p},\mathbf{q}\in\epsilon} \min(\alpha|\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) \tag{3}$$

Then, we compute scale field by solving:

$$\begin{cases} \hat{\mathbf{m}} = \arg\min_{m(\mathbf{p})\in[1,\ldots,N]} \sum_{\mathbf{p}} \min\left(||s_1(\mathbf{p}, \sigma_{m(\mathbf{p})}) - s_2(\mathbf{p} + \mathbf{w}_{\hat{m(\mathbf{p})}}(\mathbf{p}))||_1, t\right) + \sum_{\mathbf{p},\mathbf{q}\in\epsilon} \min(\beta|\sigma_{m(\mathbf{p})} - \sigma_{m(\mathbf{q})}|, \tau) \\ \hat{\sigma} = \{\sigma_{\hat{m(\mathbf{p})}}\} \end{cases} \tag{4}$$

Run the following two procedures for a desired number of iterations.

Optimize flow field $\mathbf{w}$:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{\mathbf{p}} \min\left(||s_1(\mathbf{p}, \sigma_{\hat{m(\mathbf{p})}}) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p}))||_1, t\right) + \sum_{\mathbf{p},\mathbf{q}\in\epsilon} \min(\alpha|\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) \tag{5}$$

Optimize scale field σ:

$$\begin{cases} \hat{\mathbf{m}} = \arg\min_{m(\mathbf{p})\in[1,\ldots,N]} \sum_{\mathbf{p}} \min\left(||s_1(\mathbf{p}, \sigma_{m(\mathbf{p})}) - s_2(\mathbf{p} + \hat{\mathbf{w}}(\mathbf{p}))||_1, t\right) + \sum_{\mathbf{p},\mathbf{q}\in\epsilon} \min(\beta|\sigma_{m(\mathbf{p})} - \sigma_{m(\mathbf{q})}|, \tau) \\ \hat{\sigma} = \{\sigma_{\hat{m(\mathbf{p})}}\} \end{cases} \tag{6}$$

Output: The estimated flow field $\mathbf{w}^* = \hat{\mathbf{w}}$ and scale field $\sigma^* = \hat{\sigma}$.

Figure 2. Optimization algorithm for Eqn. (2).

the threshold of scale field $\tau = 120$; the threshold of data term t is set follows the way in [12]. We run our scale-space SIFT flow following our algorithm description in Fig. 2.

4.1. Quantitative result

The original Middlebury dataset[1] images are of identical scale. To perform image matching across different scales, we re-scale images in this dataset, following the protocol in [6]. Source images are scaled to 0.7 and target images are scaled to 0.2. The ground-truths are also scaled accordingly. Quantitative image matching performance is measured by the angular error and the endpoint error which are the same as in [1]. Specifically, the angular error is the angle between estimated flow and ground truth. It is defined as $AE = \cos^{-1}\left(\frac{1.0+u*u_{GT}+v*v_{GT}}{\sqrt{1.0+u^2+v^2}\sqrt{1.0+u_{GT}^2+v_{GT}^2}}\right)$. Endpoint error is defined as $EE = \sqrt{(u-u_{GT})^2 + (v-v_{GT})^2}$. Average values and standard deviations of both angular error and endpoint error($\pm$SD) for three methods are reported in Table. 1.

Table. 1 shows that original SIFT flow fails to deal with large scale difference. Both SLS and our method can handle large scale difference and our method consistently outperforms SLS in both angular error and endpoint error measures.

We also test our method on the original Middlebury dataset. From Table 2 it is obvious that our method can perform equally well as the original SIFT flow, which has been successfully applied to scene parsing[11].

4.2. Qualitative result

To further illustrate the ability of our method, some visual comparisons are shown in this section. For our method and other compared methods, we compute dense feature correspondences between the source image and the target image; the result image is the warped source image according to the computed dense correspondences.

At first, we show the matching results as the scale factor change in Fig. 3. As shown in Fig. 3, from left to right, the scale of the source image reduces gradually. The original SIFT flow can handle small scale variation, but will inevitably fail when scale difference become large. Both our method and SLS can effectively handle large scale vari-

Data	Angular error			Endpoint error		
	SIFT flow[13]	SLS[6]	Our method	SIFT flow[13]	SLS[6]	Our method
Dimetrodon	25.99 ± 38.28	0.17 ± 0.42	$\mathbf{0.13} \pm 0.18$	59.06 ± 37.98	0.83 ± 0.40	$\mathbf{0.52} \pm 0.27$
Grove2	5.12 ± 12.89	0.15 ± 0.26	$\mathbf{0.11} \pm 0.23$	25.89 ± 39.60	0.80 ± 0.36	$\mathbf{0.48} \pm 0.34$
Grove3	5.73 ± 12.80	0.16 ± 0.32	$\mathbf{0.12} \pm 0.27$	69.24 ± 69.84	0.91 ± 0.47	$\mathbf{0.62} \pm 0.83$
Hydrangea	5.98 ± 13.82	$\mathbf{0.18} \pm 0.37$	0.26 ± 1.77	24.52 ± 34.14	0.79 ± 0.40	$\mathbf{0.63} \pm 1.18$
RubberWhale	20.73 ± 30.05	0.15 ± 0.23	$\mathbf{0.12} \pm 0.17$	63.29 ± 41.46	0.85 ± 0.50	$\mathbf{0.52} \pm 0.28$
Urban2	6.24 ± 13.32	0.32 ± 1.28	$\mathbf{0.14} \pm 0.18$	42.59 ± 55.57	1.53 ± 5.40	$\mathbf{0.65} \pm 0.53$
Urban3	9.65 ± 14.98	0.66 ± 1.39	$\mathbf{0.19} \pm 0.37$	58.25 ± 52.05	18.20 ± 35.11	$\mathbf{0.79} \pm 0.81$
Venus	5.51 ± 15.21	0.23 ± 0.47	$\mathbf{0.22} \pm 0.55$	15.94 ± 29.94	0.78 ± 0.39	$\mathbf{0.62} \pm 0.73$

Table 1. Quantitative results for SIFT flow [13], SLS [6] and our method on scaled version of Middlebury dataset [1].

Data	Angular error		Endpoint error	
	SIFT flow[13]	Our method	SIFT flow[13]	Our method
Dimetrodon	$\mathbf{9.82} \pm 8.55$	10.71 ± 11.24	$\mathbf{0.43} \pm 0.28$	0.47 ± 0.37
Grove2	8.29 ± 12.52	$\mathbf{6.65} \pm 8.37$	0.57 ± 0.64	$\mathbf{0.50} \pm 0.50$
Grove3	12.44 ± 19.18	$\mathbf{10.69} \pm 17.13$	1.10 ± 1.66	$\mathbf{0.98} \pm 1.51$
Hydrangea	8.96 ± 18.74	$\mathbf{8.23} \pm 17.53$	0.60 ± 1.17	$\mathbf{0.56} \pm 1.10$
Rubberwhale	11.46 ± 15.78	$\mathbf{10.59} \pm 14.87$	0.37 ± 0.42	$\mathbf{0.35} \pm 0.40$
Urban2	10.77 ± 17.40	$\mathbf{8.34} \pm 12.29$	1.51 ± 3.57	$\mathbf{0.77} \pm 1.74$
Urban3	14.48 ± 35.09	$\mathbf{8.28} \pm 21.29$	1.46 ± 3.23	$\mathbf{0.97} \pm 2.04$
Venus	$\mathbf{7.17} \pm 22.07$	10.93 ± 34.00	$\mathbf{0.55} \pm 1.05$	1.25 ± 4.14

Table 2. Quantitative results of original SIFT flow [13] and our method on the original version of Middlebury dataset [1].

ation.

We take the identical images presented in [6] for a fair comparison. Fig. 4 shows the case that scale difference is caused by scene motion. As shown in the figure, the original SIFT flow method manages to lock onto single scale, which causes large distortion on the result image. Both SLS and our method can find correct correspondences between the source images to the target images. Since our method do not have a region of interest (ROI) strategy as used in SLS, the warped source image becomes more flexible in deformation.

Fig. 5 shows the image matching results with scale difference in different scenes. For the flower and butterfly cases, our method can tackle scale difference to produce appealing results. For the racing car case, though each car in the source image found its correspondence on the target image, the road becomes more cluttered than the other method. So we consider it as a "failure" case.

Computational cost In this part, we compare computational time and memory cost needed by SLS and our method. According to our algorithm in Fig. 2, the computational time of our algorithm is roughly number of scale times execution time of original SIFT flow. But in SLS, computing set-to-set distance for multi-scale SIFTs is very time and memory consuming. We report average computational time and memory cost for matching one pair of images in the experiments in Sec. 4.1 in Table 3. In addition, our platform is a Xeon 3.07Ghz server with 64G memory.

	SLS [6]	Ours
Computational Time	45 minutes	2 minutes
Memory Cost	6G	1G

Table 3. Comparison of computational cost of SLS[6] and our method.

From those previous experiment, we observe that both SLS and the proposed method are able to cope with scale difference problem in matching. The significantly improved quantitative performance with much less computation burden illustrates the evident advantage of our method over SLS.

4.3. Example based color enhancement

Most successful applications of SIFT flow don't suffer from the scale problem, for example [12, 13]. In this section, we apply our algorithm to example based color enhancement. This application requires both reliable dense correspondence and the ability to handle scale change.

Due to lighting condition or camera specification, color quality of photos varies a lot. Examples are shown in Fig. 6 and Fig. 7. Traditional method uses expert and software to manually regrade color, which requires tedious work. Some recent progress [22, 19, 5] tried to use example photographs

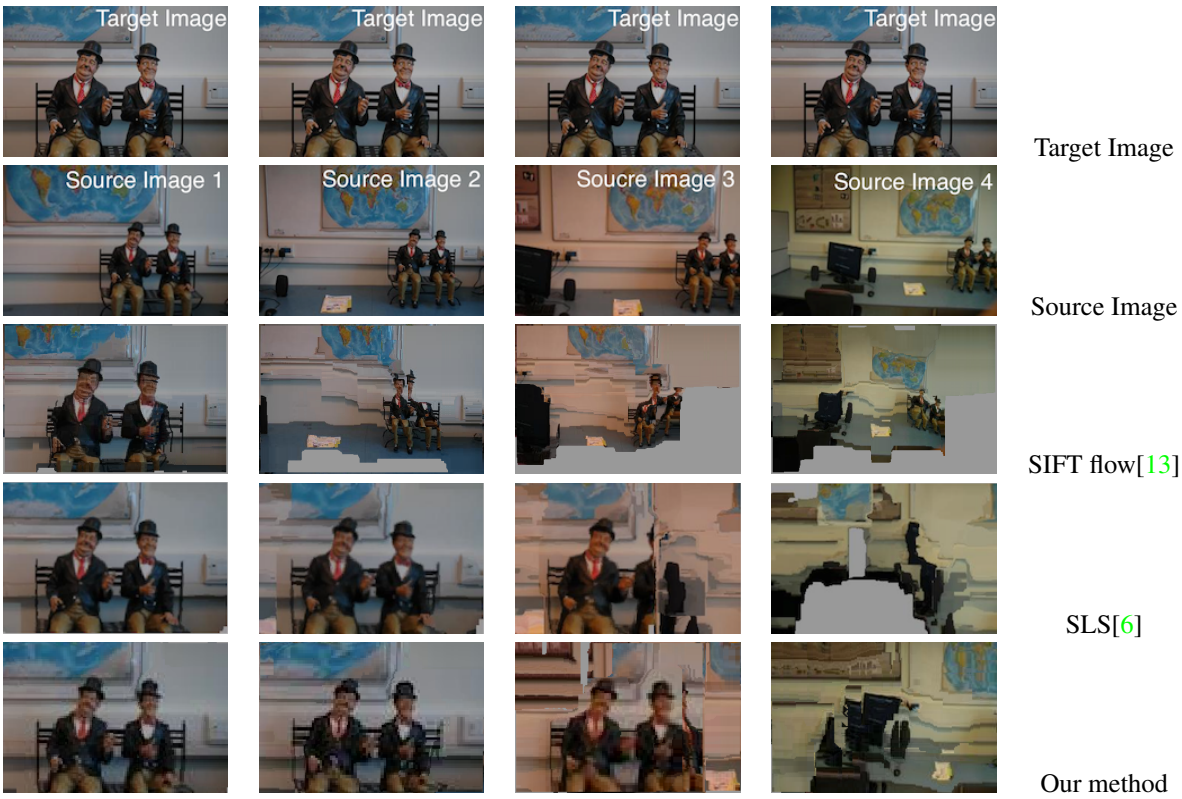

Figure 3. Matching results of SIFT flow [13], SLS [6] and our method on different scales. SIFT flow can only handle small scale variance. Our method and SLS can tolerate much bigger scale difference.

Figure 4. Matching results of SIFT flow [13], SLS [6] and our method on different scale images caused by camera and scene motion.

to regrade low quality photos' color automatically. Most of those work rely or can benefit from reliable dense correspondence of input images.

In this experiment, we use the estimated correspondent region to estimate color statistics instead of using whole image. Then we manipulate the color statistics of source image to match example image, using the color transfer technique proposed in [19]. For more detail of the color transfer algorithm, please refer to [19]. Fig. 6 shows the difference between using correspondent region and whole image as input of color transfer. The large background region in example image makes the color statistics quite different, though these two input images share a large portion of content. We can not generate satisfactory result based on SIFT flow's correspondence, because its limitation of handling scale change.

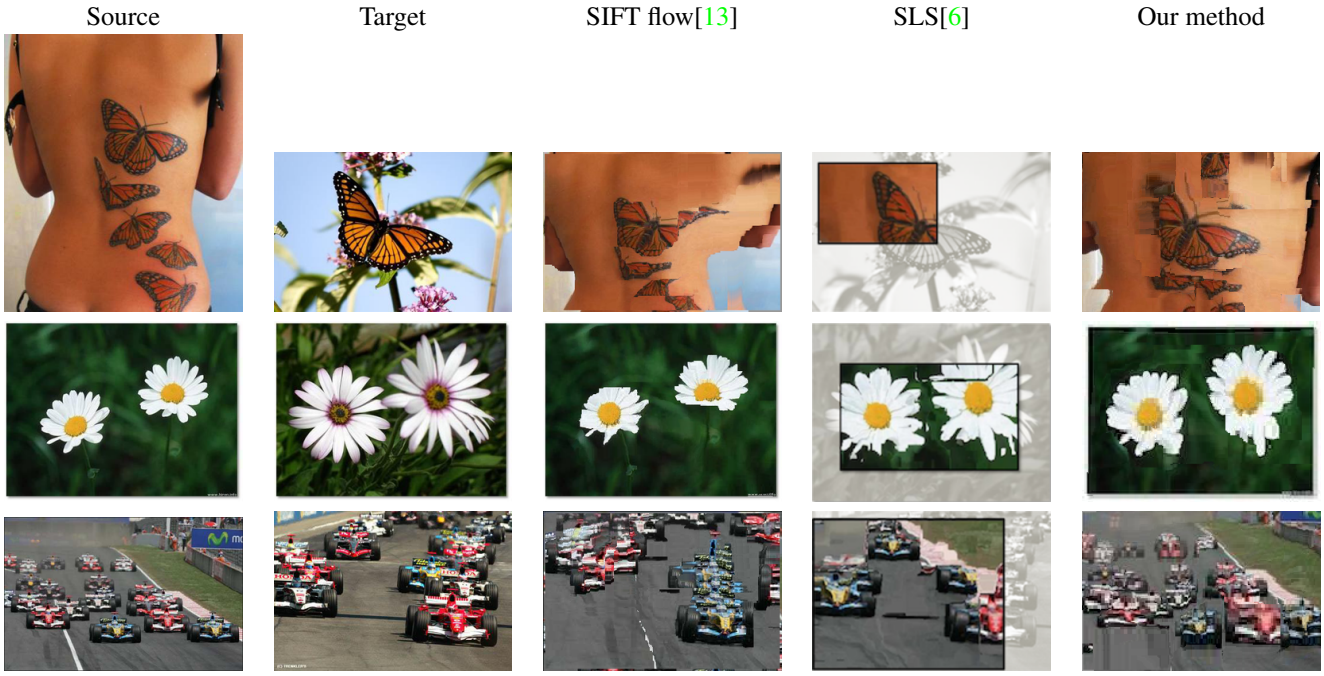

Figure 5. Matching results of SIFT flow [13], SLS [6] and our method on multi-scale images in different scenes.

In Fig. 7, we show another case of example based enhancement. In this case, source image is captured by a low-end android phone and example image is retrieved by google image search. We are not able to find an example image which shares the same content with our low quality images, but we can easily find visually similar exemplar images. Computing correspondence for this case is difficult and [5] failed to get any correspondent region. Our method can still produce reasonable result. For the second image in Fig. 7, the correspondence is not reliable, but the color statistics of correspondent region and whole image are still similar. In this situation, our color enhancement method shows similar behaviour to [19].

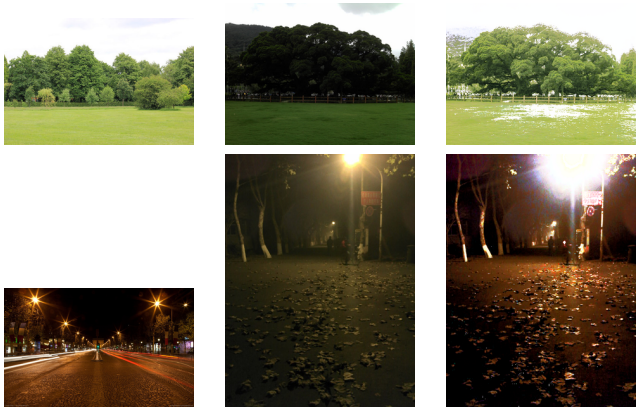

Figure 7. Our method can enhance image based on visually similar example, while NRDC[5] will fail at finding correspondence and can not produce result

5. Conclusion

In this paper, we have presented a general image matching algorithm, which extends from SIFT flow by automatically exploring the scale-space of the dense SIFT features. Our system is simple, intuitive, easy to reproduce[1], and very effective. By explicitly estimating the scale field, our method can effectively deal with large scale difference in the image pairs, while reaching the similar performance as the original SIFT flow method in normal images.

The computational complexity and memory consumption of our method is linear to SIFT flow, which makes it easy to scale to deal with a large number images. We expect our method to be applied in a variety of computer vision tasks for scene understanding, retrieval, and 3D construction.

[1]Source code can be found at http://weichaoqiu.com/code/ssf.zip

Acknowledgement

This work is supported by the National Natural Science Foundation of China (grant No. 61222308) and the NSF under Grants IIS-1216528 (IIS-1360566), IIS-0844566(IIS-1360568) and IIS-0917141.

Example Image | Source Image | Our method | Pitie et al.[19] | NRDC[5]

Figure 6. Comparison with Pitié et al.[19] and NRDC[5]. The dense correspondent region can provide more reliable color statistics than whole image

References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 3, 4, 5

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, 2002. 1

[3] J. Crowley. *A representation for visual information*. PhD thesis, Carnegie Mellon University, 1981. 2

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006. 3

[5] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Non-rigid dense correspondence with applications for image enhancement. In *ACM Transactions on Graphics (TOG)*, volume 30, page 70. ACM, 2011. 5, 7, 8

[6] T. Hassner, V. Mayzels, and L. Zelnik-Manor. On sifts and their scales. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June. 2012. 1, 3, 4, 5, 6, 7

[7] I. Kokkinos and A. Yuille. Scale invariance without scale selection. In *CVPR*, pages 1–8, 2008. 2

[8] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006. 1

[9] T. Lindeberg. Scale-space theory in computer vision. *Kluwer Academic Publishers*, 1994. 1

[10] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998. 2

[11] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1972–1979. IEEE, 2009. 4

[12] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2368–2382, 2011. 1, 4, 5

[13] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011. 1, 2, 3, 5, 6, 7

[14] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: dense correspondence across different scenes. In *ECCV*, pages 28–42. Springer, 2008. 2

[15] H. Liu and S. Yan. Common visual pattern discovery via spatially coherent correspondences. In *CVPR*, pages 1609–1616. IEEE, 2010. 2

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1, 2

[17] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004. 1, 2

[18] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990. 1

[19] F. Pitié, A. C. Kokaram, and R. Dahyot. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding*, 107(1):123–137, 2007. 5, 6, 7, 8

[20] M. Rubinstein, C. Liu, and W. T. Freeman. Annotation propagation in large image databases via dense image correspondence. In *ECCV*, pages 85–99. Springer, 2012. 1

[21] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010. 1

[22] B. Wang, Y. Yu, and Y.-Q. Xu. Example-based image color and tone style enhancement. *To appear in ACM TOG*, 30:4. 5

[23] A. Witkin. Scale space filtering. In *Proc. of 8th International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983. 1

[24] A. Yuille and T. Poggio. Scaling theorems for zero crossings. *IEEE PAMI*, 1:15–25, 1986. 1

[25] J. Zhao, J. Ma, J. Tian, J. Ma, and D. Zhang. A robust method for vector field learning with application to mismatch removing. In *CVPR*, pages 2977–2984. IEEE, 2011. 2

[26] F. Zhou and F. De la Torre. Factorized graph matching. In *CVPR*, pages 127–134. IEEE, 2012. 2