

## FIND: A Neat Flip Invariant Descriptor\*

Xiaojie Guo and Xiaochun Cao

School of Computer Science and Technology, Tianjin University, China  
 {xguo, xcao}@tju.edu.cn

### Abstract

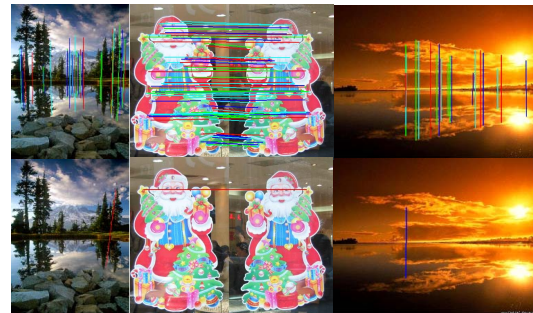
In this paper, we introduce a novel Flip INvariant Descriptor (FIND). FIND improves the degenerated performance resulted from image flips and reduces both space and time costs. Flip invariance of FIND enables the intractable flip detection to be achieved easily, instead of duplicately implementing the procedure. To alleviate the pressure brought by the increasing scale of image and video data, FIND utilizes a concise structure with less storage space. Comparing to SIFT, FIND reduces 35.94% length for a descriptor. We compare FIND against SIFT with respect to accuracy, speed and space cost. An application to image search over a database of 3.27 million descriptors is also shown.

### 1. Introduction

In the field of visual data retrieval, multimedia copy detection methods have received significant attention and achieved numerous breakthroughs in last decades. Existing works can be generally divided into text-based or content-based. The text-based methods used by most of commercial search-engines can search data quickly but lead to ambiguous and noisy results. Unlike text-based methods, the content-based ones [10] are more intelligent to search targets. Most of the content-based methods are based on local image descriptors, such as SIFT [7] and SURF [1], to extract image features as content information. However, none of these descriptors can handle the flip detection effectively since they are not flip invariant. One common solution [4] is to perform an extra flipped query, *i.e.* querying the flipped video sequence, which is apparently inefficient.

Other important factors in visual data retrieval are the time and space complexities, as the scale of image and video data increases rapidly. Although new algorithms such as re-ranking algorithm [5], BoF [11], HE

\*This work was supported by NSFC (No. 60905019, 50735003), SRF for ROCS, SEM, Open Projects Program of NLPR, Tianjin University 985 research fund, and SKL of PMTI.



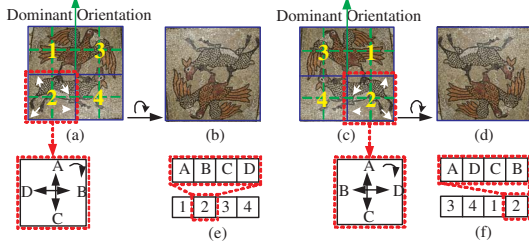
**Figure 1.** Matching result comparison. Top row: FIND matching results. Bottom row: SIFT matching results.

and WGC [4], are able to speed up the search significantly, the length of the descriptor is still an important factor for an efficient search.

In this work, we propose a novel flip invariant descriptor. We achieve the flip invariance by introducing a new cell ordering scheme. As shown in Fig. 1, our method finds quite a few correct matches while SIFT fails in the presence of flip. In addition, FIND is tolerant to scale, rotation and affine transformations as SIFT does. More importantly, a FIND descriptor uses a concise structure, which is 35.94% shorter than a SIFT one.

#### 1.1 Related Work

Local image descriptors are crucial for many applications such as visual retrieval [4], copy detection [9], symmetry detection [6] and action recognition [8]. To obtain discriminative features, the detection is the determinant and various detection schemes have been proposed, such as Difference-of-Gaussian (DoG) detector [7] and Harris corner detector [3]. FIND adopts DoG due to its robustness and distinctiveness. The scale space of a source image is constructed to detect extrema. Then relative high thresholds for DoG and Hessian matrix are adopted to eliminate the candidates with low contrast and edge response respectively.



**Figure 2.** Example of different image flips and the corresponding SIFT descriptors. A simplified SIFT descriptor with  $2 \times 2$  cells and 4 oriented bins in each cell is used for illustration. (a) The original image. (b)-(d) The combined , horizontal and vertical flip versions of (a). (e)-(f) The SIFT descriptors for (a) and (c).

## 2 Our Local descriptor

### 2.1 Image flips

Here, we focus on the image flips, and therefore translation, similarity, affine and perspective transformations are out of the scope of this paper. Suppose the rotating center of an image is the point  $\mathbf{p}_c = (w/2, h/2)$ , where  $w$  and  $h$  are the width and height of the image. The flipped image and the original image are related by a  $3 \times 3$  matrix  $\mathbf{H}$  denoted as

$$\mathbf{H}(\theta, \epsilon) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (1)$$

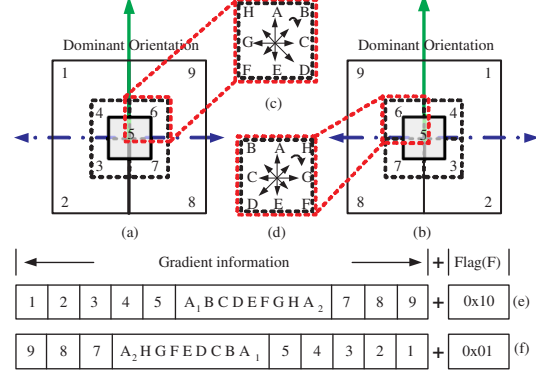
where  $\epsilon$  is the flip coefficient and  $\theta$  is the rotation angle. It is easy to verify that the vertical and horizontal flips can be generated by setting  $\epsilon$  to  $-1$  while varying  $\theta$ . For example, the transformations  $\mathbf{H}_h$  and  $\mathbf{H}_v$  for the horizontal and the vertical flips (Fig. 2 (c) and (d)) are

$$\mathbf{H}_h = \mathbf{H}(0, -1), \quad \mathbf{H}_v = \mathbf{H}(\pi, -1). \quad (2)$$

In other words, horizontal and vertical flips are equivalent by rotating the coordinate system by  $\pi$ . Interestingly, the combination of the vertical and horizontal flips, *i.e.*  $\mathbf{H}_c = \mathbf{H}_v \mathbf{H}_h = \mathbf{H}(\pi, 1)$ , is equal to the original as shown in Fig. 2 (b). Therefore, all flips can be treated as the horizontal flip ( Fig. 2 (c)) after rotation.

As shown in Fig. 2 (c), only the column order is reversed for the flipped image. Simultaneously, the gradient orientations in the interest regions with and without flip are symmetric about the dominant orientation while the magnitudes are approximately<sup>1</sup> unchanged, *e.g.* the white arrows in the  $2^{nd}$  cells of Fig. 2 (a) and (c). As a

<sup>1</sup>Noises or distortions might influence the gradients



**Figure 3.** Illustration of FIND in the situations with and without flip. (a) A keypoint and its interest region in the original image. (b) The flipped version of (a). (c)-(d) Distribution of eight orientations in the  $6^{th}$  cells of (a) and (b). (e)-(f) FIND descriptors for (a) and (b).

result, SIFT encodes the cells and the oriented orientations in a fixed pattern, as shown in Fig. 3 (e) and (f), which however are different to each other.

In the following section, we will present our flip invariant descriptor FIND according to the relationship of the flips described above.

### 2.2 FIND

We detail FIND by four steps: i) splitting strategy, ii) weighting strategy, iii) descriptor organization and iv) extra flag.

**Splitting strategy.** As shown in Fig. 3, each keypoint of FIND is supported by a square interest region. Before elaborating the interest region of FIND, we define a splitting strategy called overlap-extension (O-E) strategy. That is the cells at outer layers contain and extend the ones at inner layers. O-E considers the information around the keypoint rather than splitting the interest region without attention on the spatial relationship of the keypoint. As shown in Fig. 3 (a) and (b), the interest regions are split into 9 cells according to O-E. The  $3^{rd}$ ,  $4^{th}$ ,  $6^{th}$  and  $7^{th}$  cells ( $C_{L2}$ ) contain and extend the central cell labeled as 5 ( $C_{L1}$ ). The same relation holds true for the  $1^{st}$ ,  $2^{nd}$ ,  $8^{th}$  and  $9^{th}$  cells ( $C_{L3}$ ) and the remainders ( $C_{L1}$  and  $C_{L2}$ ).

**Weighting strategy.** Intuitively, pixels close to the keypoint in the interest region shall support the feature more significantly than distant ones. Hence, it is reasonable to weight the pixels according to the distances to the keypoint. The Gaussian weighted window satisfies this requirement. Moreover, O-E makes overlapped pixels multi-weighted to emphasize the pixels near the keypoint. According to our interest region design as shown in Fig. 3 (a) and (b),  $C_{L3}$  is four times the size

of the remaining cells but holds less importance. In our experiments, the weight of  $C_{L3}$  is half of its original weight from the Gaussian weighted window, since we use  $L_2$  distance to compare descriptors.

**Descriptor organization.** To explain the structure of FIND descriptor clearly, we divide the organizing strategy into two steps. The first step is to organize 9 cells in the interest region. FIND organizes 9 cells in the transverse ‘S’ pattern rather than simply in column- or row-major-order. As shown in Fig. 3 (a), the cell sequence is [123456789], and [987654321] in Fig. 3 (b). Although the cell orders in the situations with and without flip are not identical, they are reversible. Therefore, an extra flag is able to indicate the choices of descriptor sequences. The second step is the organization of oriented gradients. Each cell of FIND consists of 8 oriented gradients, *i.e.*  $ABCDEFGH$  in Fig. 3 (c) and (d). For preserving the reversibility of the 81 elements, we use 9 elements, in our descriptor, to store them,  $A_1BCDEFGHA_2$ , where  $A_1 = A_2 = A/\sqrt{2}$ , *e.g.* Fig. 3 (e) and (f), no matter the image is flipped or not. Moreover, we use  $L_2$  distance to match the descriptors and this signature satisfies  $A_1^2 + A_2^2 = A^2$ . We also normalize 81 elements to make FIND tolerant to illumination changes.

**Extra flag.** To obtain the identical descriptor for the feature in the situations with and without flips, FIND uses an additional flag to indicate the order of the FIND signature in the matching step. Similar with MIFT [2], we employ gradient information as the criterion since FIND itself is gradient-based:

$$m_r = \sum_{k=j}^{N_{bin}/2-j} L_{(n_d-k+N_{bin})\%N_{bin}}, \quad (3)$$

$$m_l = \sum_{k=j}^{N_{bin}/2-j} L_{(n_d+k+N_{bin})\%N_{bin}}, \quad (4)$$

where  $N_{bin}$  is the number of orientation bins,  $n_d$  is the dominant orientation index,  $L_i$  is the gradient magnitude in the  $\frac{i \times 2\pi}{N_{bin}}$  direction, and  $j$  is an integer with the value range  $[1, \lfloor N_{bin}/4 \rfloor]$ . According to this measurement, we adaptively label the flag based on the relationship between  $m_l$  and  $m_r$ . As shown in Fig. 3 (a) and (b), the left-pointing and right-pointing blue dashed arrows mean  $m_l$  and  $m_r$ . Nevertheless,  $m_l$  and  $m_r$  might be close to each other due to symmetry, noises, lighting variations and other factors. We use two extra bits to serve as a flag (F) that is used to indicate the descriptor orders, as shown in Fig. 3 (e) and (f). To be more robust in such situations, the flag is operated by bitwise OR operation with  $0x10$  ( $0x01$ ) when  $m_l$  ( $m_r$ ) is greater than  $\tau \max\{m_l, m_r\}$ , where  $\tau$  is a threshold fell in between

0 and 1. In the matching process, the bitwise AND operation between F and  $0x10$  ( $0x01$ ) is operated to certify whether the positive (negative) sequence is used.

Finally, a complete FIND contains  $81 + 1 = 82$  elements as shown in Fig. 3 (e) and (f) and has two possible sequences for the descriptor.

## 3 Experiments

### 3.1 Descriptor testing

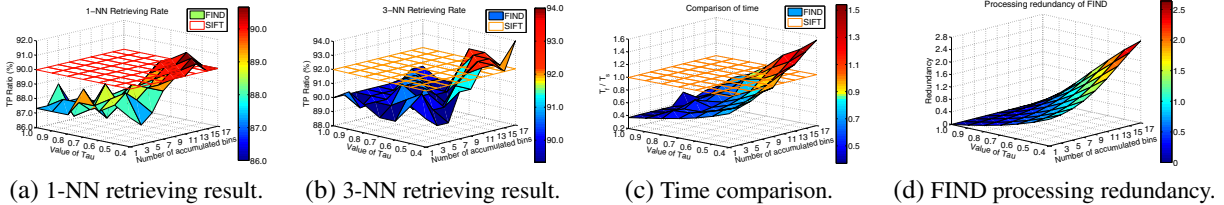
The parameters,  $j$  and  $\tau$ , affect the FIND descriptor’s performance such as the speed and space. Intuitively, a smaller  $\tau$  makes FIND to be more robust to noises and distortions, with the price of extra comparisons in the matching process. To find the optimal parameters, we apply FIND in the visual search application, and use the INRIA Holidays dataset<sup>2</sup> as our test data. The images in this dataset are under different viewing conditions, and therefore are appropriate for the evaluation of FIND. We compare FIND against SIFT with respect to accuracy, speed and storage space.

To eliminate other factors that might influence the comparison results between FIND and SIFT, we use direct comparing method. For SIFT, queries need to be processed twice: once for the original queries and the other for the flipped ones. We select 500 images from the dataset as our database and 150 images as query set, wherein 1/3 queries have been vertically flipped and 1/3 horizontally flipped. There are 63 different combinations of the number of accumulated orientations (9 options) and  $\tau$  (7 options) in total.

Figure 4 shows experimental results in which the number of accumulated orientation bins and  $\tau$  are varied. As shown in Fig. 4 (a), the retrieving ratio is 86.00% when  $\tau = 1$ , however, the results continue to improve up to 90.67% as  $\tau$  drops to 0.4. The trend of the 3-NN results is broadly similar to the 1-NN one, although the 3-NN ratios are from 89.33% to 94.00%. For the 1-NN and 3-NN ratios, SIFT retrieves 90.00% and 92.00% respectively. Note that the influence of the number of accumulated orientation bins ( $j$ ) on the accuracy is not in evidence.

The time expenditure of FIND with different parameters is examined in Fig. 4 (c). The graph shows that lower  $\tau$  and more accumulated bins lead to more processing time. Since each FIND has two possible cases of descriptor sequences, the comparing processes of FIND may, at most, be 3 times more than SIFT. Combining both redundancy and time expenditure, Fig. 4 (c) and (d) reveal that the time is directly in proportion to the redundancy. Although a smaller  $\tau$  and a greater  $j$

<sup>2</sup><http://lear.inrialpes.fr/~jegou/data.php>



**Figure 4.** Comparison between FIND with different parameters and SIFT. Mesh presents the performances of SIFT and surface presents those of FIND.

lead to a higher computational cost, the advantage is a higher accuracy. FIND runs at most 0.5413 times slower than SIFT and obtains targets more accurately. However, most time expenditures of FIND are lower than SIFT while the accuracies descend slightly. The least time that FIND costs, in our experiments, is 0.3707 times the length of time that SIFT uses with 4.00% and 2.67% accuracy losses of the 1-NN and 3-NN ratios.

With respect to the storage space, FIND is significantly lower than SIFT and unchanged under different parameters. FIND spends 449.34M to store 1.436494  $\times 10^6$  descriptors for the images in the database, which is 63.73% of the size that SIFT uses. Due to the duplicate computation, SIFT uses 422.66M to store the descriptors for the queries, while FIND costs 134.67M.

### 3.2 Application to image search

We implement the direct comparing method based on FIND using more images. We choose 1070 images from INRIA Holidays dataset as our database and 340 queries as the query set. There are 1/3 queries that are vertically flipped and 1/3 horizontally flipped at random. We choose four different parameter combinations (number of accumulated bins,  $\tau$ ), including (1, 1.0), (9, 0.9), (17, 0.8) and (17, 0.4).

To show the comparison clearly, we choose SIFT results as our baseline in terms of time and space. SIFT uses 1.047486  $\times 10^6$  seconds to process the queries and costs 1.5613 GB (D) + 1.0247 GB (Q) to store the descriptors. Table 1 demonstrates the trade off between accuracy and cost, *i.e.* more accurate results need more time to process.  $F_{(1,1.0)}$  reduces the time to 0.3825 times of SIFT with 2.23% and 1.47% accuracy losses for 1-NN and 3-NN results. Comparing to SIFT,  $F_{(17,0.4)}$  costs as much as 1.7491 times of SIFT to increase the 1-NN and 3-NN retrieving ratios by 1.88% and 1.76% respectively.

## 4 Conclusion

Our proposed FIND provides robustness and convenience for visual data search. Its compact size signif-

**Table 1.** Performance comparison.

	1-NN	3-NN	Time	Space <sub>D</sub>	Space <sub>Q</sub>
$F_{(1,1.0)}$	0.862	0.885	0.383	0.637	0.319
$F_{(9,0.9)}$	0.871	0.885	0.489	0.637	0.319
$F_{(17,0.8)}$	0.871	0.897	0.709	0.637	0.319
$F_{(17,0.4)}$	0.903	0.918	1.749	0.637	0.319
<b>SIFT</b>	0.884	0.900	1.000	1.000	1.000

icantly reduces the storage space and makes descriptors matching efficient, especially for mass data. We have compared it against SIFT with respect to accuracy, speed and space expenditure. The results have demonstrated that FIND with different parameters is effective and flexible to different demands. Future work will aim at optimizing the current algorithm and applying FIND on symmetry object detection and copy detection.

## References

- [1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: speeded up robust features. In *ECCV*, pages 404–417, 2006.
- [2] X. Guo and X. Cao. Mift: A mirror reflection invariant feature descriptor. In *ACCV*, volume 2, pages 536–545, 2009.
- [3] C. Harris and M. J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 20, pages 147–152, 1988.
- [4] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, pages 304–317, 2008.
- [5] Y. Jing and S. Baluja. Pagerank for product image search. In *Proc. of WWW*, pages 307–316, 2008.
- [6] S. Lee and Y. Liu. Curved glide-reflection symmetry detection. In *CVPR*, pages 1046–1053, 2009.
- [7] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.
- [8] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, pages 357–360, 2007.
- [9] X. Wu, Y. Zhang, Y. Wu, J. Guo, and J. Li. Invariant visual patterns for video copy detection. In *ICPR*, pages 1–4, 2008.
- [10] T. Yeh, J. Lee, and T. Darrell. Photo-based question answering. In *ACM MM*, pages 389–398, 2008.
- [11] X. Zhou, X. Zhuang, S. Yan, S. Chang, M. Johnson, and T. Huang. Sift-bag kernel for video event analysis. In *ACM MM*, pages 229–238, 2008.