A Novel Real-Time Local Visual Feature for Omnidirectional Vision Based on FAST and LBP

Huimin Lu, Hui Zhang, and Zhiqiang Zheng

College of Mechatronics Engineering and Automation,
National University of Defense Technology, Changsha, China
{lhmnew,huizhang_nudt,zqzheng}@nudt.edu.cn

Abstract. A novel real-time local visual feature, namely FAST+LBP, is proposed in this paper for omnidirectional vision. It combines the advantages of two computationally simple operators by using Features from Accelerated Segment Test (FAST) as the feature detector and Local Binary Patterns (LBP) operator as the feature descriptor. The matching experiments of the panoramic images from the COLD database are performed to determine its best parameters, and to evaluate and compare its performance with SIFT. The experimental results show that our algorithm performs better, and features can be extracted in real-time.

1 Introduction

In comparison with global visual features, local visual features have better discriminative power, and are more robust to occlusion. Furthermore, good local visual features can be invariant to image rotation, image translation, image scale, changes of view, and even changes of illumination. Thus local visual features have become increasingly popular in recent years, and they have been applied very well in many computer/robot vision problems, such as image retrieval, image stitching, wide baseline matching, object recognition, place recognition, texture recognition, robot localization, and robot navigation. Local visual feature algorithm consists of feature detector and feature descriptor, and lots of algorithms have been proposed. In feature detector, Harris [1], Susan [2], DOG [3], MSER [4], Harris-Laplace [5], Harris-Affine [5], Hessian-Affine [5], Features from Accelerated Segment Test (FAST) [6], etc. have been designed; in feature descriptor, SIFT [3], SURF [7], GLOH [8], Local Binary Patterns (LBP) [9], etc. have been used to calculate the feature vectors over the feature region. Many researchers have done lots of work on evaluation and comparison of these algorithms [8][10][11][12], while relative source codes (or binaries), and many image databases have been released for further evaluation with new algorithms.

Although local visual features have so many advantages, a common deficiency for most of the existing algorithms is that their computation costs are usually high. This deficiency limits the actual application of local visual features, especially in those situations with high real-time requirements. Therefore several improved versions of the above algorithms have been proposed to accelerate the

J. Ruiz-del-Solar, E. Chown, and P.G. Ploeger (Eds.): RoboCup 2010, LNAI 6556, pp. 291–302, 2011.

feature detection or description. Fast approximated SIFT is presented in Ref. [13]. Compared to standard SIFT, it uses a box filter to calculate the DoM (Difference-of-Mean) images efficiently based on integral images. The key-points can then be detected, and the descriptor is also accelerated by using an integral orientation histogram. The experiments show speed increases by a factor of eight while the performance is only slightly decreased. In the iterative SIFT [14], the number of features can be defined in advance, so the process of searching the key-points continues iteratively without the need for sequentially going through the whole scale space. When it is applied in robot's localization, the computation effort of the feature extraction and matching process can be reduced as much as possible while high localization accuracy can be maintained. SURF [7] also takes the advantage of integral images. In feature detector, SURF approximates second order Gaussian derivatives with box filters, and image convolutions with these box filters can be computed rapidly by using integral images. In feature descriptor, Haar wavelet responses, which also can be quickly computed via integral images, are used to construct the descriptor vector. The experimental results show that the performance of SURF outperforms its competitors.

Because omnidirectional vision can provide a 360° view of the robot's environment in an image, it has become more and more popular as a visual sensor for robots. In this paper, we will propose a novel real-time local visual feature for omnidirectional vision which can be applied in the actual engineering problems with high real-time requirements. Features from Accelerated Segment Test (FAST) [6] will be used as the feature detector, and Local Binary Patterns (LBP) [9] as the feature descriptor, so an algorithm named FAST+LBP will be designed. The panoramic images in the COLD database [15] will be used to test our algorithm. The following sections are organized as follows: FAST and LBP are introduced in section 2; our FAST+LBP is proposed in section 3; the best algorithm parameters are determined by experiments, and the performance of FAST+LBP is evaluated and compared with standard SIFT in section 4; section 5 is the conclusion of this paper.

2 FAST and LBP

2.1 FAST

The corner feature is defined in FAST detector by the following Segment-Test algorithm: If more than N contiguous pixels in a Bresenham circle of radius r around a center pixel p are all brighter than p by some threshold or all darker than p by some threshold, there is a corner feature at p. Machine learning is utilized to speed up this corner detection process. Every pixel has 16 attributes corresponding to the 16 pixels in the Bresenham circle (for $r = 3$), and each attribute can be 0, 1 or -1. If a pixel with position x on the circle of p is brighter(darker) than p, the corresponding attribute is 1(-1). Otherwise, the attribute is 0. A decision tree can be learned by using ID3 to select the pixels in the circle which yield the most information about whether the center pixel is a corner. So a pixel can be classified as a corner feature or not more efficiently,

which means the Segment-Test algorithm is accelerated. This decision tree is then converted into C-code, creating a long string of nested if-then-else statements which is compiled and used as a corner detector. Finally non-maximal suppression is applied to remove corners which have an adjacent corner with higher value of the sum of the absolute difference between the pixels in the circle and the center pixel.

For N values of 9, 10, 11, and 12, the corresponding FAST algorithms are named FAST 9, FAST 10, FAST 11, and FAST 12. According to the experiments in Ref. [6], FAST 9 seems to be the best FAST detector, and it is over five times faster than the quickest non-FAST detector. The FAST algorithm also significantly outperforms Harris, DoG, Harris-Laplace, SUSAN, etc. in repeatability, except in cases with large amounts of added image noise.

2.2 LBP Method

LBP is firstly proposed as texture operator [16][17]. It describes each pixel by the relative gray values of its neighboring pixels. If the gray value of the neighboring pixel is higher or equal to that of the center pixel, the binary value is set to be one, otherwise to be zero. The LBP value of a center pixel in (x, y) position can be calculated over the neighborhood as follows:

$$LBP_{R,N}(x,y) = \sum_{i=0}^{N-1} s(n_i - n_c)2^i, \quad s(t) = \begin{cases} 1, t \geq 0 \\ 0, \text{otherwise} \end{cases} \tag{1}$$

where n_c is the gray value of the center pixel, and n_i the gray value of N equally spaced pixels on a circle of radius R. According to Eq.(1), the $LBP_{R,N}$ value may be any integer between 0 and $2^N - 1$. The histogram of the $LBP_{R,N}$ values computed over an image region (the histogram dimension will be 2^N) can be used for texture description, and it has been proven to be robust against illumination changes. It is also very fast to compute.

Several modified versions of LBP method have been described in Ref. [17] for achieving rotation invariance and reducing the histogram dimension of LBP. When the image is rotated, the gray value n_i will correspondingly move along the perimeter of the circle around n_c, so different $LBP_{R,N}$ may be calculated. To remove the effect of rotation, the first modified version with rotation invariance is defined as follows:

$$LBP^{ri}_{R,N}(x,y) = min\{ROR(LBP_{R,N}, i) \quad | \quad i = 0, 1, ..., N-1\} \tag{2}$$

where $ROR(LBP_{R,N}, i)$ performs a circular bit-wise right shift on the N-bit number $LBP_{R,N}$ i times. $LBP^{ri}_{R,N}$ can have 36 different values when $N = 8$, and the histogram dimension of $LBP^{ri}_{R,N}$ over an image region is 36.

In the second version named as *uniform* LBP, at most two one-to-zero or zero-to-one transitions in the circular binary code are allowed, so whether a LBP is uniform can be judged by the following definition:

$$U(LBP_{R,N}) = |s(n_{N-1} - n_c) - s(n_0 - n_c)| + \sum_{i=1}^{N-1} |s(n_i - n_c) - s(n_{i-1} - n_c)| \tag{3}$$

If $U(LBP_{R,N}) \leq 2$, the LBP is uniform. The *uniform* LBP, expressed as $LBP^{u2}_{R,N}$, can have $N(N-1)+2$ different values, so the histogram dimension of $LBP^{u2}_{R,N}$ over an image region is $N(N-1)+2+1$ (the final 1 corresponds to those *non-uniform* LBP).

The third version is the *uniform* LBP with rotation invariance which combines the above two modifications. So $LBP^{riu2}_{R,N}$ value is computed as follows:

$$LBP^{riu2}_{R,N}(x,y) = \begin{cases} \sum_{i=0}^{N-1} s(n_i - n_c), & U(LBP_{R,N}) \leq 2 \\ N+1, & \text{otherwise} \end{cases} \tag{4}$$

$LBP^{riu2}_{R,N}$ value can have $N+1+1$ different values, so the histogram dimension of $LBP^{riu2}_{R,N}$ over an image region is $N+1+1$.

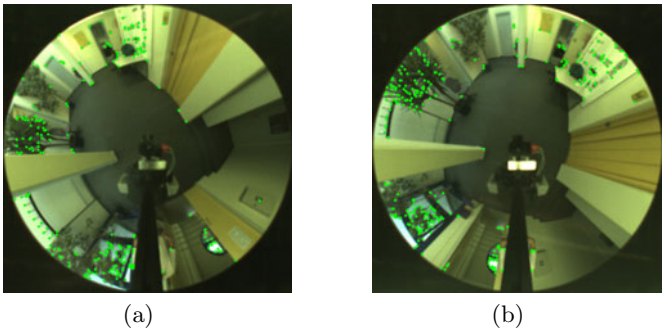
All the three modified LBP versions can be considered to be a mapping from the original LBP with wide value range to the corresponding modified LBP with narrow value range, so the histogram dimension can be reduced with different extents. In practice, the mapping process is implemented by a look-up table which can be created in advance according to the different mapping mode.

3 Our Novel Real-Time Local Visual Feature

Our novel real-time local visual feature, namely FAST+LBP, is divided into three steps: feature detector, feature region determination, and feature descriptor.

3.1 FAST Feature Detector

Because the FAST 9 algorithm has a low computation cost and excellent performance in repeatability, it was chosen as the feature detector. The typical panoramic images and the corner features detected by FAST 9 are demonstrated in Fig.1. The images are from the COLD database [15]. The two images are captured by the robot's omnidirectional vision in two different positions. The robot's translation is 0.7561 m, and the rotation is 0.9053 rad.



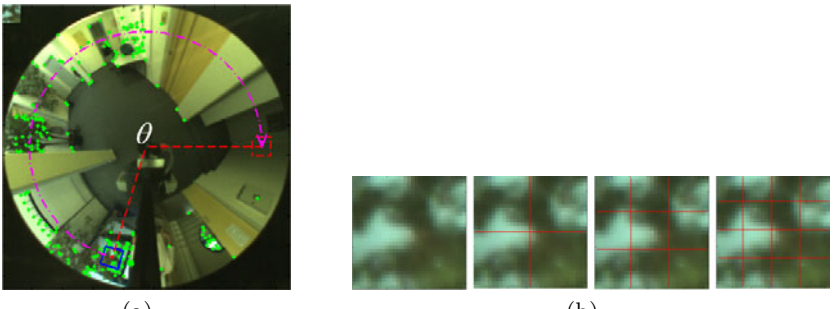
(a) (b)

Fig. 1. A pair of panoramic images from the COLD database and the feature detecting results by FAST 9. The green points are the detected corner features.

3.2 Feature Region Determination

After a corner feature has been detected, a surrounding image region should be determined, and then a descriptor can be extracted from the image region. Some affine invariant feature detectors [5] have been proposed to adapt the feature region to affine transformations by iterative algorithms. Although they can provide better performance, the computation complexity increases significantly [5]. Therefore we do not consider affine invariance for our real-time local visual feature algorithm. We adopt the feature region determining method proposed in Ref. [18]. Rectangular image regions surrounding corner features are firstly determined in the radial direction, and then rotated to a fixed orientation, as shown in Fig.2. Fig.2(a) shows how a determined feature region is rotated to the fixed orientation. During the rotation process, bilinear interpolation is used.

In the next section we will compare this feature region determining method with the one which determines feature regions directly in horizontal and vertical directions through experimentation. The size of each feature region is also an important parameter, and the best size will be determined in the next section.



(a) (b)

Fig. 2. (a) The blue rectangle is a feature region, and it is rotated by angle θ to a fixed orientation. The small region on the top left of the image is the rotated feature region. (b) The different grids that the feature region can be divided into. From left to right: 1×1 cell, 2×2 cells, 3×3 cells, 4×4 cells.

3.3 Feature Descriptor with LBP

The final step of local visual feature algorithm is to describe the features by calculating vectors according to the information of feature regions. Recently, LBP has been used as feature descriptor in Ref. [9], and the strength of SIFT descriptor is also combined. The SIFT-like grid is used, but SIFT gradient features are replaced by LBP-based features. In this paper, we use the same approach to extract descriptors for the detected features by FAST.

A LBP value for each pixel of the feature region can be calculated according to the introduction in section 2.2. In order to incorporate spatial information into the descriptor, the feature region can be divided into different grids such as 1×1 cell, 2×2 cells, 3×3 cells, 4×4 cells, as shown in Fig.2(b). For each cell,

the histogram of LBP values is created, and then all the histograms are concatenated into a vector as the descriptor. Finally, the descriptor is normalized to unit length. The descriptor dimension is $M \times M \times$*the histogram dimension* for $M \times M$ cells. Therefore, the resulting descriptor is a 3D histogram of LBP feature locations and LBP values. In calculating the histogram, the LBP values can be weighted with a Gaussian window overlaid over the whole feature region, or with uniform weights over the whole region. The latter means that the feature weighting is omitted.

The performance and dimension of LBP descriptor will be affected greatly by different algorithm parameters such as the number of cells, different R and N, Gaussian or uniform weighting, LBP mode including the original LBP, LBP^{ri}, LBP^{u2}, and LBP^{riu2} as introduced in section 2.2. The best parameters will be determined by experiments in the next section.

4 Experimental Evaluation and Discussion

In this section, we will introduce the experimental setup firstly, and then determine the best parameters for FAST+LBP by experiments. The performance and the needed computation time of our algorithm will be compared with SIFT.

4.1 Experimental Setup

COLD is a freely available database which provides a large-scale, flexible testing environment for robot's vision-based topological localization [15]. The panoramic images are captured by the same omnidirectional vision in different rooms and under various light conditions. We will use the typical panoramic images and image series to perform our experiments.

When local visual features are applied in robot's localization, robot's SLAM, etc., the features should be matched between the image pairs captured in different imaging conditions, such as different robot positions and various lighting conditions. Therefore we evaluate the performance of local visual feature according to the feature matching results. For each feature descriptor in an image, We calculate its Euclidean distances with all the feature descriptors in another image needing to be matched. We consider that a match is found between the feature pair with the closest distance if the ratio of the closest to second-closest distance is smaller than threshold T_{ratio} [3] as follows:

$$ratio = \frac{the\ closest\ distance}{the\ second-closest\ distance} \leq T_{ratio} \tag{5}$$

In this paper, we will evaluate the overall performance of local visual feature as a whole, but not evaluate the detector and descriptor independently as in Ref. [5][6][8][9]. Therefore we use *matching score versus* $1-precision$ as the criterion for performance evaluation, instead of *recall versus* $1-precision$ which is used

to evaluate the descriptor's performance in Ref. [8][9]. We define *matching score* in the same way as Ref. [10]:

$$matching\ score = \frac{the\ number\ of\ correct\ matches}{the\ smaller\ number\ of\ features\ in\ the\ pair\ of\ images} \tag{6}$$

We define $1 - precision$ as follows in the same way as Ref. [8][9]:

$$1 - precision = \frac{the\ number\ of\ false\ matches}{the\ number\ of\ correct\ matches + the\ number\ of\ false\ matches} \tag{7}$$

After the feature matching is finished, an 18 bin histogram is created from $\triangle\theta_i = normalize(\theta_i - \theta_i^{'})$ using all the matched features, where θ_i and $\theta_i^{'}$ are the rotated angles of the ith pair of matched features relative to the fixed orientation in section 3.2, and $normalize(.)$ means normalizing the angle to $[0, 2\pi)$. According to the character of omnidirectional vision, the relative angle of each pair of correctly matched features, namely φ, should be almost the same, so it can be estimated by computing the mean value of those $\triangle\theta_i$ falling into the highest bin, and φ is the rotation angle of robot approximately. If $|\triangle\theta_i - \varphi| < T_{angle}$, where T_{angle} is the threshold determined by experiments, the match related to $\triangle\theta_i$ is a correct match. Otherwise, it is a false match.

As we change the threshold T_{ratio}, the curve of *matching score versus* $1 - precision$ can be acquired to evaluate the performance of algorithms.

4.2 Parameter Evaluation for FAST+LBP

The evaluation of different parameter settings for FAST+LBP is carried out in this experiment to determine the best parameters. As presented in section 3, six parameters will affect the performance of FAST+LBP. We will test their different settings as follows:

The size of the feature region: 15×15, 19×19, 23×23, 27×27, 31×31, 35×35, 39×39, 43×43 pixels; the feature region determining method: method 1–determining the feature's rectangular region directly in horizontal and vertical directions, method 2–determining the rectangular region in the radial direction and then rotating it to a fixed orientation as proposed in section 3.2; the number of grids: 1×1 cell, 2×2 cells, 3×3 cells, 4×4 cells; the N and R: $N = 8$ and $R = 1$, $N = 16$ and $R = 2$, $N = 24$ and $R = 3$; the LBP mode: the original LBP, LBP^{ri}, LBP^{u2}, LBP^{riu2}; the weighting strategy: Gaussian weighting, uniform weighting.

Because of a huge amount of different combinations of above parameters, only one parameter is varied at a time while the others are kept fixed. The pair of images in Fig.1 are used to perform the feature matching, and the curves of *matching score versus* $1 - precision$ with different parameters are shown in Fig.3. The red curves in Fig.3 represent the best parameters. From the matching results, we see that 27×27 pixels for the feature region, region determining method 2, 2×2 cells, $N = 8$, $R = 1$, LBP^{u2}, and Gaussian weighting provide the best performance for FAST+LBP. The descriptor dimension of our final FAST+LBP is $2\times2\times(8 \times 7 + 2 + 1)$=236, as shown in Fig.4.

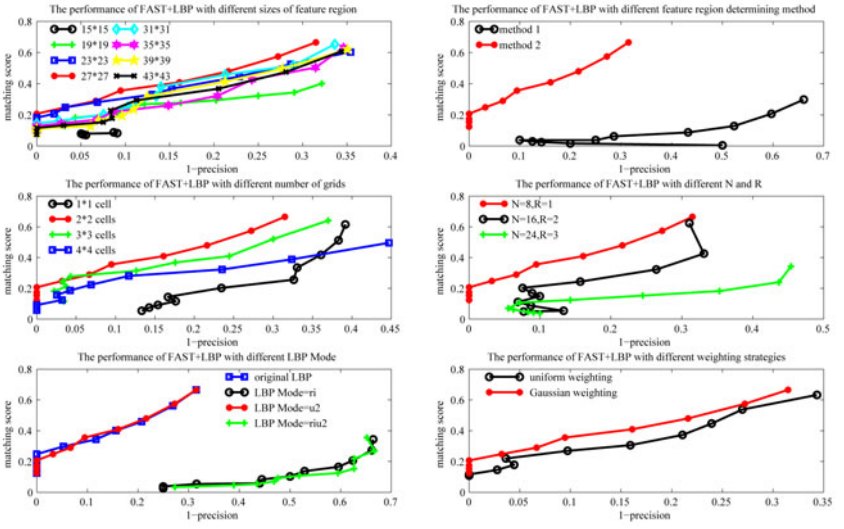
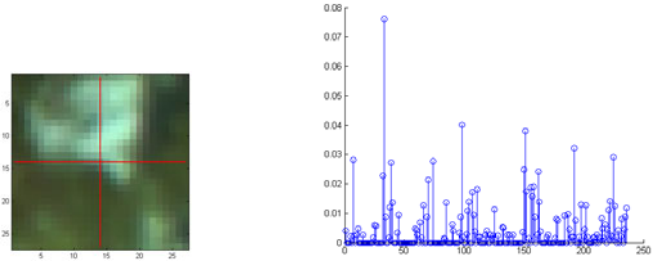


Fig. 3. Parameter evaluation results for FAST+LBP. Only one parameter is varied at a time while the others are kept fixed with the best parameters.

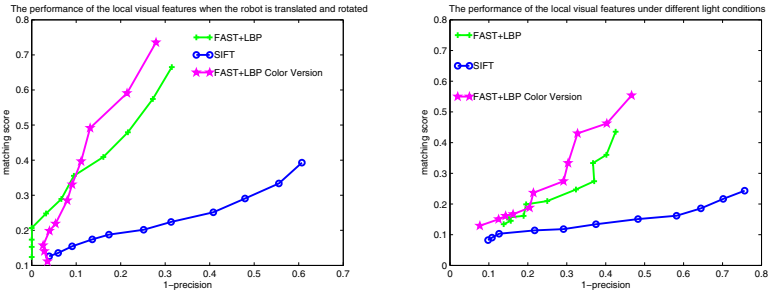


(a) (b)

Fig. 4. Our final FAST+LBP. (a) The scale-up feature region which is divided into 2×2 cells. (b) The resulting feature descriptor.

4.3 Performance Comparison of FAST+LBP and SIFT

The performance comparison of FAST+LBP and SIFT is carried out in this experiment. The SIFT we adopt is implemented by Andrea Vedaldi [19]. Because most of the current robot's cameras are color ones, and the images in the COLD database are color images, we also compare the color version of FAST+LBP together. In our color version of FAST+LBP, the feature detector still uses the gray values of images, but the descriptor is computed in all of the R, G, B color channel, so its dimension is three times of that of the gray version. Two pairs of images are used. The first one is that in Fig.1, and they are acquired when the



(a) (b)

Fig. 5. The performance comparison of FAST+LBP, color version of FAST+LBP, and SIFT. (a) Robot is translated and rotated. (b) Under different lighting conditions.

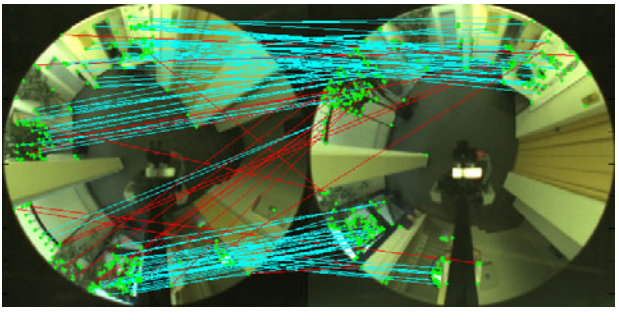


Fig. 6. The matching results of the pair of images in Fig.1 by FAST+LBP. The cyan lines represent the correct matches, and the red lines represent the false matches.

robot is translated and rotated. The second pair of images are captured when the robot is in the same position but under different lighting conditions. The matching results of these two pairs of images are depicted in Fig.5 (a) and (b) respectively.

We fix the threshold T_{ratio} as 0.95 after making a compromise between *matching score* and *precision*. The matching result of the pair of images in Fig.1 by FAST+LBP with this threshold is shown in Fig.6. Then we can evaluate how *matching score* changes with the different imaging conditions of omnidirectional vision caused by the robot's translation, rotation, and different lighting conditions. Three image series are used in this evaluation. The first one includes 30 images which are acquired as the robot is only translated. The translation increases with the image number, and the maximal translation is 1.7975 m. The second one includes 17 images which are acquired as the robot is only rotated. The rotation increases with the image number, and the maximal rotation is π. The third one includes 5 images which are acquired in the same position and under different lighting conditions. We perform the feature matching between

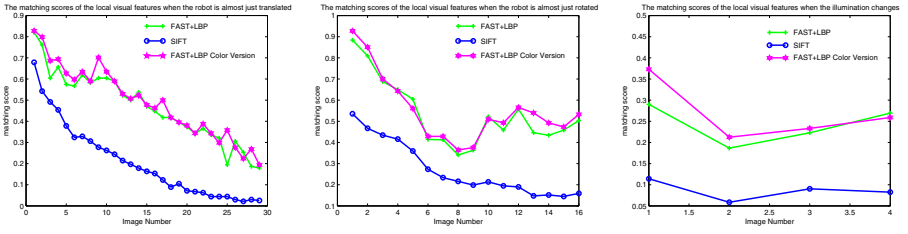


Fig. 7. The *matching score* with different imaging conditions by FAST+LBP, color version of FAST+LBP, and SIFT. (left) The robot is only translated. (middle) The robot is only rotated. (right) Under different lighting conditions.

the first image and all the other images in each series, so how *matching score* changes with different imaging conditions is acquired, as shown in Fig.7.

From the above experimental results, we clearly see that FAST+LBP provides better performance than SIFT in image matching, and it is a excellent local visual feature for omnidirectional vision. The matching results are not bad even when the robot is translated and rotated greatly and the lighting conditions are very different. So FAST+LBP is robust to rotation, different lighting conditions, and robot's certain translation. The color version seems a little better than the gray version. However, its computation cost is much higher, because the descriptor of the color version should be computed in each of the three color channels. Furthermore, it takes much more time to match features for the color version because of the larger descriptor dimension. So we prefer the gray version rather than the color version.

4.4 Comparison of the Needed Computation Time

In this experiment, we collect 125 panoramic images from the COLD database, and then extract local visual features from these images using FAST+LBP and SIFT respectively. Our FAST+LBP is implemented by C++, and the SIFT we use is implemented by C++ and Matlab using C-Mex technique [19]. The computer is equipped with 2.26GHz Duo CPU and 1.0G memory. The number of features, the time needed to extract all the features in an image, and the average time needed to extract one feature are demonstrated in Fig.8. The time needed in the three steps of FAST+LBP is also shown. We see that FAST+LBP extracts about 150∼350 features per image, less than SIFT, but it can be performed much faster. After doing statistics on the computation time, we find that the time needed to extract all the features by SIFT in an image is 508 times that of FAST+LBP, and the average time needed to extract one feature by SIFT is 115 times that of FAST+LBP. The computation time needed to extract all the features in an image by FAST+LBP is from 5ms to 20ms, so it can be performed in real-time.

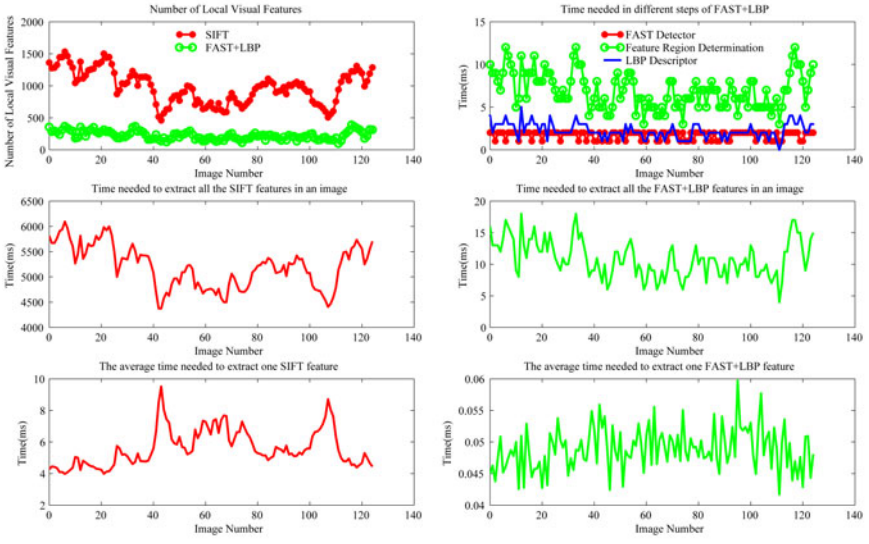


Fig. 8. The comparison of the needed computation time by FAST+LBP and SIFT

5 Conclusions

A novel local visual feature, namely FAST+LBP, is proposed for omnidirectional vision. It combines the advantages of two computationally simple operators by using FAST as the feature detector and LBP operator as the feature descriptor. The best algorithm parameters were determined by experiments. The comparisons between FAST+LBP, color version of FAST+LBP, and SIFT were performed, and the experimental results show that our algorithm has better performance than SIFT, and features can be extracted in real-time. Our local visual feature can be applied to computer/robot vision tasks with high real-time requirements.

Acknowledgement

We thank Edward Rosten and Tom Drummond for their release of the FAST source code, Marko Heikkilä and Timo Ahonen for their release of the LBP source code, Andrea Vedaldi for his release of the SIFT source code, and Andrzej Pronobis, Barbara Caputo, et al. for providing their COLD database.

References

1. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of Alvey Vision Conference, pp. 147–151 (1988)
2. Smith, S.M., Brady, J.M.: SUSAN-a new approach to low level image processing. Int. J. Comput. Vision 23(1), 45–78 (1997)

3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (2004)
4. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. In: Proceedings of BMVC, pp. 384–393 (2002)
5. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. Int. J. Comput. Vision 60(1), 63–86 (2004)
6. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)
7. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding 110, 346–359 (2008)
8. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. 27(10), 1615–1630 (2005)
9. Heikkilä, M., Pietikäinen, M., Schmid, C.: Description of interest regions with local binary patterns. Pattern Recognition 42, 425–436 (2009)
10. Mikolajczyk, K., Tuytelaars, T., Schmid, C., et al.: A comparison of affine region detectors. Int. J. Comput. Vision 65(1), 43–72 (2005)
11. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. Int. J. Comput. Vision 37(2), 151–172 (2000)
12. Li, J., Allinson, N.M.: A comprehensive review of current local features for computer vision. Neurocomputing 71, 1771–1787 (2008)
13. Grabner, M., Grabner, H., Bischof, H.: Fast approximated SIFT. In: Narayanan, P.J., Nayar, S.K., Shum, H.-Y. (eds.) ACCV 2006. LNCS, vol. 3851, pp. 918–927. Springer, Heidelberg (2006)
14. Tamimi, H., Andreasson, H., Treptow, A., et al.: Localization of Mobile Robots with Omnidirectional Vision using Particle Filter and Iterative SIFT. Robotics and Autonomous Systems 54, 758–765 (2006)
15. Pronobis, A., Caputo, B.: COLD: The Cosy Localization Database. The International Journal of Robotics Research 28(5), 588–594 (2009)
16. LBP website, `http://www.ee.oulu.fi/mvg/page/lbp_bibliography`
17. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. 24(7), 971–987 (2002)
18. Andreasson, H., Treptow, A., Duckett, T.: Self-Localization in non-stationary environments using omni-directional vision. Robotics and Autonomous Systems 55, 541–551 (2007)
19. Vedaldi, A.: SIFT source code, `http://www.vlfeat.org/~vedaldi/code/sift.html`