21^{st} *Computer Vision Winter Workshop*
Luka Čehovin, Rok Mandeljc, Vitomir Štruc (eds.)
Rimske Toplice, Slovenia, February 3–5, 2016

Quantitative Comparison of Feature Matchers Implemented in OpenCV3

Zoltan Pusztai
Eötvös Loránd University
Budapest, Hungary
puzsaai@inf.elte.hu

Levente Hajder
MTA SZTAKI
Kende u. 13-17. Budapest, Hungary-1111
http://web.eee.sztaki.hu

Abstract. *The latest V3.0 version of the popular Open Computer Vision (OpenCV) framework has just been released in the middle of 2015. The aim of this paper is to compare the feature trackers implemented in the framework. OpenCV contains both feature detector, descriptor and matcher algorithms, all possible combinations of those are tried. For the comparison, a structured-light scanner with a turntable was used in order to generate very accurate ground truth (GT) tracking data. The tested algorithm on tracking data of four rotating objects are compared. The results is quantitatively evaluated as the matched coordinates can be compared to the GT values.*

1. INTRODUCTION

Developing a realistic 3D approach for feature tracker evaluation is very challenging since realistically moving 3D objects can simultaneously rotate and translate, moreover, occlusion can also appear in the images. It is not easy to implement a system that can generate ground truth (GT) data for real-world 3D objects. The Middlebury database[1] is considered as the state-of-the-art GT feature point generator. The database itself consists of several datasets that had been continuously developed since 2002. In the first period, they generated corresponding feature points of real-world objects [23]. The first Middlebury dataset can be used for the comparison of feature matchers. Later on, this stereo database was extended with novel datasets using structured-light [24] or conditional random fields [18]. Even subpixel accuracy can be achieved in this way as it is discussed in [22].

However, the stereo setup is too strict limitation for us, our goal is to obtain tracking data via multiple frames.

[1] http://vision.middlebury.edu/

The description of the optical flow datasets of Middlebury database was published in [3]. It was developed in order to make the optical flow methods comparable. The latest version contains four kinds of video sequences:

1. *Fluorescent images*: Nonrigid motion is taken by both color and UV-camera. Dense ground truth flow is obtained using hidden fluorescent texture painted on the scene. The scenes are moved slowly, at each point capturing separate test images in visible light, and ground truth images with trackable texture in UV light.

2. *Synthesized database*: Realistic images are generated by an image syntheses method. The tracked data can be computed by this system as every parameters of the cameras and the 3D scene are known.

3. *Imagery for Frame Interpolation.* GT data is computed by interpolating the frames. Therefore the data is computed by a prediction from the measured frames.

4. *Stereo Images of Rigid Scenes.* Structured light scanning is applied first to obtain stereo reconstruction. (Scharstein and Szeliski 2003). The optical flow is computed from ground truth stereo data.

The main limitation of the Middlebury optical flow database is that the objects move approximately linearly, there is no rotating object in the datasets. This is a very strict limitation as tracking is a challenging task mainly when the same texture is seen from different viewpoint.

It is interesting that the Middlebury multi-view database [25] contains ground truth 3D reconstruction of two objects, however, the ground truth track-

ing data were not generated for these sequences. Another limitation of the dataset is that only two low-textured objects are used.

It is obvious that tracking data can also be generated by a depth camera [26] such as Microsoft Kinect, but its accuracy is very limited. There are other interesting GT generators for planar objects such as the work proposed in [8], however, we would like to obtain the tracked feature points of real spatial objects.

Due to these limitations, we decided to build a special hardware in order to generate ground truth data. Our approach is based on a turntable, a camera, and a projector. They are not too costly, but the whole setup is very accurate as it is shown in our accepted paper [19].

2. Datasets

We have generated four GT datasets as it is published in our mentioned paper [19]. The feature points are always selected by the tested feature generator method in all frames and then these feature locations are matched between the frames. Then the matched point are filtered: the fundamental matrix [9] is robustly computed using 8-point method with RANSAC for every image pair and the outliers are removed from the results. The method implemented in the OpenCV framework is used for this robustification.

Examples for the moving GT feature points of the generated sets are visualized in Figures 1– 4. Point locations are visualized by light blue dots.

The feature matchers are tested in four data sequences:

- **Dinosaur**. A typical computer vision study deals with the reconstruction of a dinosaurs as it is shown in several scientific papers, e.g [6]. It has a simple diffuse surface that is easy to reconstruct in 3D, hence the feature matching is possible. For this reason, a dino is inserted to our testing dataset.

- **Flacon.** The plastic holder is another smooth and diffuse surface. A well-textured label is fixed on the surface.

- **Plush Dog.** The tracking of the feature point of a soft toy is a challenging task as it does not have a flat surface. A plush dog is included into the testing database that is a real challenge for feature trackers.

- **Poster.** The last sequence of our dataset is a rotating poster in a page of a motorcycle magazine. It is a relatively easy object for feature matchers since it is a well-textured plane. The pure efficiency of the trackers can be checked in this example due to two reasons: (i) there is no occlusion, and (ii) the GT feature tracking is equivalent to the determination of plane-plane homographies.

Figure 5. Reconstructed 3D model of testing objects. Top: Plush Dog. Center: Dinosaur. Bottom: Flacon.

2.1. GT Data Generation

Firstly, the possibilities is overviewed that OpenCV can give about feature tracking. These are the currently supported feature detectors in OpenCV AGAST [13], AKAZE [17], BRISK [10], FAST [20], GFTT [28] (Good Features To Track – also known as Shi-Tomasi corners), KAZE [2], MSER [14], ORB [21].

However, if you compile the contrib(nonfree) repository with the OpenCV, you can also get the

Figure 1. GT moving feature points of sequence 'Flacon'.

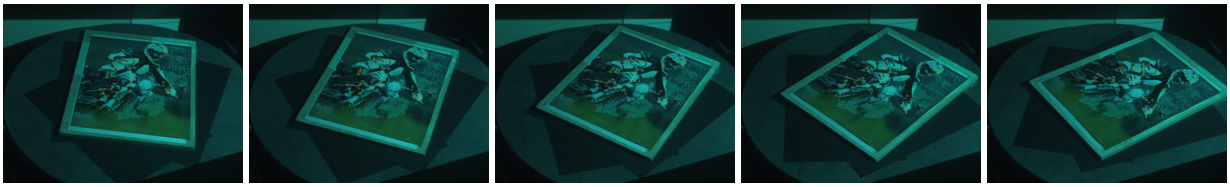

Figure 2. GT moving feature points of sequence 'Poster'.

following detectors: SIFT [12], STAR [1], and SURF [4].

We use our scanner to take 20 images about a rotating object. After each image taken, a structured light sequence is projected in order to make the reconstruction available for every position. (reconstructing only the points in the first image is not enough.)

Then we start searching for features in these images using all feature detectors. After the detection is completed, it is required to extract descriptors. Descriptors are needed for matching the feature points in different frames. The following descriptors are used (each can be found in OpenCV): AKAZE [17], BRISK [10], KAZE [2], ORB [21]. If one compiles the contrib repository, he/she can also get SIFT [12], SURF [4], BRIEF [5], FREAK [16], LATCH [11], DAISY [27] descriptors [2].

Another important issue is the parameterization of the feature trackers. It is obvious that the most accurate strategy is to find the best system parameters for the methods, nevertheless the optimal parameters can differ for each testing video. On the other hand, we think that the authors of the tested methods can set the parameters more accurately than us as they are interested in good performance. For this reason, the default parameter setting is used for each method, and we plan to make the dataset available for everyone and then the authors themselves can parameterize their methods.

After the detection and the extraction are done, the matching is started. Every image pair is taken into consideration, and match each feature point in the first image with one in the second image. This means that every feature point in the first image will have a pair in the second one. However, there can be some feature locations in the second image, which has more corresponding feature points in the first one, but it is also possible that there is no matching point.

The matching itself is done by calculating the minimum distances between the descriptor vectors. This distance is defined by the feature tracking method used. The following matchers are available in OpenCV:

- L_2 – *BruteForce*: a brute force minimization algorithm that computes each possible matches. The error is the L_2 norm of the difference between feature descriptors.
- L_1 – *BruteForce*: It is the same as L_2 – *BruteForce*, but L_1 norm is used instead of L_2 one.
- *Hamming* – *BruteForce*: For binary feature descriptor (BRISK, BRIEF, FREAK, LETCH,ORB,AKAZE), the Hamming distance is used.
- *Hamming2* – *BruteForce*: A variant of the hamming distance is used. The difference between Hamming and Hamming2 is that the former considers every bit as element of the vector, while Hamming2 use integer number, each bit pair forms a number from interval $0 \ldots 3$ [3].

[2]The BRIEF descriptor is not invariant to rotation, however, we hold it in the set of testing algorithms as it surprisingly served good results.

[3]OpenCV's documentation is not very informative about

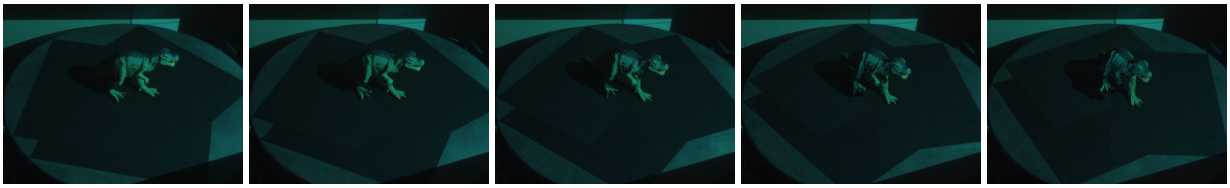

Figure 3. GT moving feature points of sequence 'Dinosaur'.

Figure 4. GT moving feature points of sequence 'Plush Dog'.

- *Flann-Based*: FLANN (Fast Library for Approximate Nearest Neighbors) is a set of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features [15].

It is needed to point out that one can pair each feature detector with each feature descriptor but each feature matchers is not applicable for every descriptor. An exception is thrown by OpenCV if the selected algorithms cannot work together. But we try to evaluate every possible selection.

The comparison of the feature tracker predictions with the ground truth data is as follows: The feature points are reconstructed first in 3D using the images and the structured light. Then, because it is known that the turntable was rotated by 3 degrees per images, the projections of the points are calculated for all the remaining images. These projections were compared to the matched point locations of the feature trackers and the L_2 norm is used to calculate the distances.

3. Evaluation Methodology

The easiest and usual way for comparing the tracked feature points is to compute the summa and/or average and/or median of the 2D tracking errors in each image. This error is defined as the Euclidean distance of the tracked and GT locations. This methodology is visualized in Fig. 6.

Hamming2 distance. They suggest the usage of that for ORB features. However, it can be applied for other possible descriptors, all possible combinations are tried during our tests.

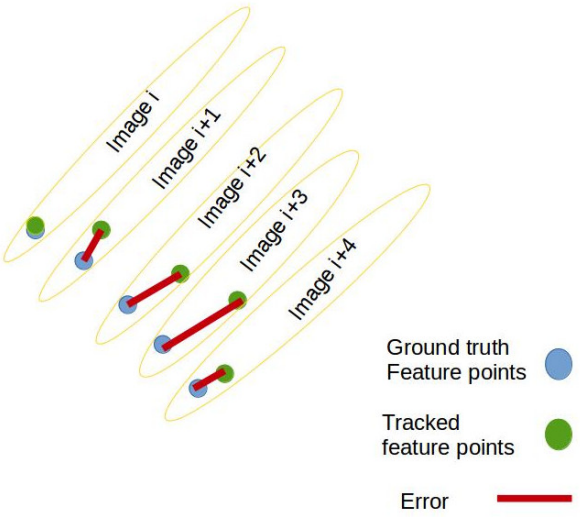

Figure 6. Error measurement based on simple Euclidean distances.

However, this comparison is not good enough because if a method fails to match correctly the feature points in an image pair, then the feature point moves to an incorrect location in the next image. Therefore, the tracker follows the incorrect location in the remaining frames and the new matching positions in those images will also be incorrect.

To avoid this effect, a new GT point is generated at the location of the matched point even if it is an incorrect matching. The GT location of that point can be determined in the remaining frames since that point can be reconstructed in 3D as well using the structured light scanning, and the novel positions of the new GT point can be determined using the calibration data of the test sequence.

Then the novel matching results are compared to

all the previously determined GT points. The obtained error values are visualized in Fig. 7.

The error of a feature point for the i-th frame is the weighted average of all the errors calculated for that feature. For example, there is only one error value for the second frame as the matching error can only be compared to the GT location of the feature detected in the first image. For the third frame, there are two GT locations since GT error generated on both the first (original position) and second (position from first matching) image. For the i-th image, $i-1$ error values are obtained. the error is calculated as the weighted average of those. It can be formalized as

$$Error_{p_i} = \sum_{n=1}^{i-1} \frac{||p_i - p'_{i,n}||_2}{i-n} \qquad (1)$$

where $Error_{p_i}$ is the error for the i-th frame, p_i the location of the tested feature detector, while $p'_{i,n}$ is the GT location of the feature points reconstructed from the n-th frame. The weights of the distances is $1/(i-n)$ that means that older GT points has less weights. Remark that the Euclidean (L_2) norm is chosen in order to measure the pixel distances.

If a feature point is only detected in one image and was not being followed in the next one (or was filtered out in the fundamental-matrix-based filtering step), then that point is discarded.

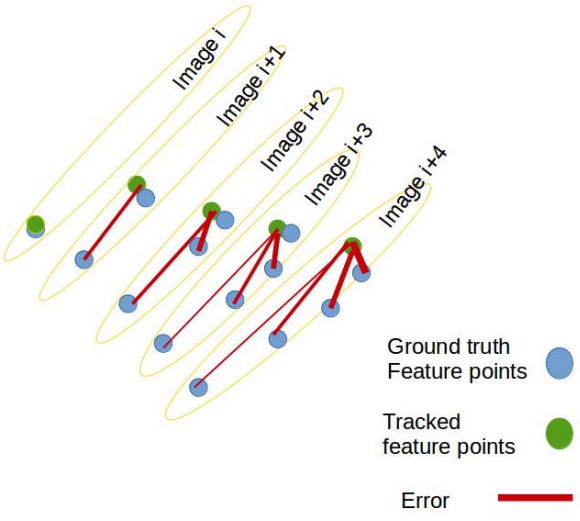

Figure 7. Applied error measurement.

After the pixel errors are valuated for each point in all possible images, the minimum, maximum, summa, average, and median error values of every feature points are calculated per image. The number of tracked feature points in the processed image is also counted. Furthermore, the average length of the feature tracks is calculated which shows that in how many images an average feature point is tracked through.

4. Comparison of the methods

The purpose of this section is to show the main issues occurred during the testing of the feature matchers. Unfortunately, we cannot show to the Reader all the charts due to the lack of space.

General remark. The charts in this section show different combinations of detectors, descriptors, and matchers. The method 'xxx:yyy:zzz' denotes in the charts that the current method uses the detector 'xxx', descriptor 'yyy', and matcher algorithm 'zzz'.

4.1. Feature Generation and Filtering using the Fundamental Matrix

The number of the detected feature points is examined first. It is an important property of the matcher algorithms since many good points are required for a typical computer vision application. For example, at least hundreds of points are required to compute 3D reconstruction of the observed scene. The matched and filtered values are calculated as the average of the numbers of generated features for all the frames as features can be independently generated in each image of the test sequences. Tables 1– 4 show the number of the generated features (left) and that of the filtered ones.

There are a few interesting behaviors within the data:

- The best images for feature tracking are obtained when the poster is rotated. The feature generators give significantly the most points in this case. It is a more challenging task to find goof feature points for the rotating dog and dinosaur. It is because the area of these objects in the images are smaller than that of the other two ones (flacon and poster).

- It is clearly seen that number of SURF feature points are the highest in all test cases after outlier removal. This fact suggests that they will be the more accurate features.

- The MSER method gives the most number of feature points, however, more than 90% of those are filtered. Unfortunately, the OpenCV3 library does not contain sophisticate matchers for

Table 1. Average of generated feature points and inliers of Sequence 'Plush Dog'.

Detector	#Features	#Inliers
BRISK	21.7	16.9
FAST	19.65	9.48
GFTT	1000	38.16
KAZE	68.6	40.76
MSER	**5321.1**	10.56
ORB	42.25	34.12
SIFT	67.7	42.8
STAR	7.15	5.97
SURF	514.05	**326.02**
AGAST	22.45	11.83
AKAZE	144	101.68

Table 2. Average of generated feature points and inliers of Sequence 'Poster'.

Detector	#Features	#Inliers
BRISK	233.55	188.79
FAST	224.75	139.22
GFTT	956.65	618.75
KAZE	573.45	469.18
MSER	**4863.6**	40.29
ORB	259.5	230.76
SIFT	413.35	343.08
STAR	41.25	35.22
SURF	1876.95	**1577.73**
AGAST	275.75	200.25
AKAZE	815	761.4

MSER such as [7], therefore its accuracy is relatively low.

- Remark that the GFTT algorithm usually gives 1000 points as the maximum number was set to thousand for this method. It is a parameter of OpenCV that may be changed, but we did not modify this value.

4.2. Matching accuracy

Two comparisons were carried out for the feature tracker methods. In the first test, every possible combination of the feature detectors and descriptors is examined, while the detectors are only combined with their own descriptor in the second test.

Table 3. Average of generated feature points and inliers of Sequence 'Flacon'.

Detector	#Features	#Inliers
BRISK	219.7	160.99
FAST	387.05	275.4
GFTT	1000	593.4
KAZE	484.1	387.93
MSER	**3664.1**	31.72
ORB	337.65	287.49
SIFT	348.15	260.91
STAR	69.1	54.86
SURF	952.95	**726.83**
AGAST	410.15	303.45
AKAZE	655	553.11

Table 4. Average of generated feature points and inliers of Sequence 'Dinosaur'.

Detector	#Features	#Inliers
BRISK	21.55	14.8
FAST	51.05	27.01
GFTT	1000	92
KAZE	58.55	33.92
MSER	**5144.4**	17.86
ORB	67.1	45.87
SIFT	52.8	34.96
STAR	3.45	3.45
SURF	276.95	**132.61**
AGAST	55	29.86
AKAZE	89.1	59.2

It is important to note that not only the errors of feature trackers should be compared, we must also pay attention to the number of features in the images and the length of the feature tracks. A method with less detected features usually obtains better results (lower error rate) than other methods with higher number of features. The mostly used chart is the AVG-MED, where the average and the median of the errors are shown.

Testing of all possible algorithms.

As it is seen in Fig 8 (sequence 'Plush Dog'), the SURF method dominates the chart. With the usage of SURF, DAISY, BRIEF, and BRISK descriptors more than 300 feature points remained and the median values of the errors are below 2.5 pixels, while the average is around 5 pixels. Moreover, the points are tracked through 4 images in average which yields pretty impressive statistics for the SURF detector.

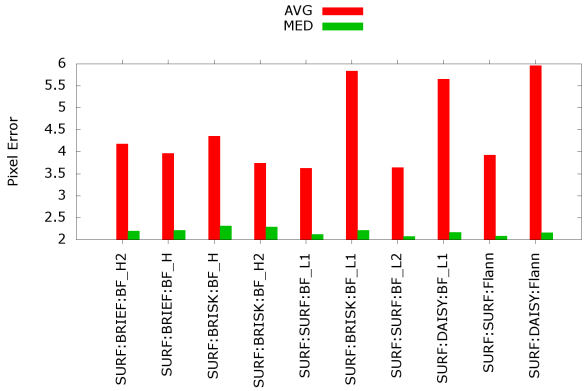

Figure 8. Average and median errors of top 10 methods for sequence 'Plush Dog'.

The next test object was the 'Poster'. The results are visualized in Fig 4.2. It is interesting to note that if the trackers are sorted by the number of the outliers and plot the top 10 methods, only the AKAZE detector remains where more than 90 percent of the feature points was considered as inlier. Besides the high number of points, average pixel error is between 3 and 5 pixels depending on the descriptor and matcher type.

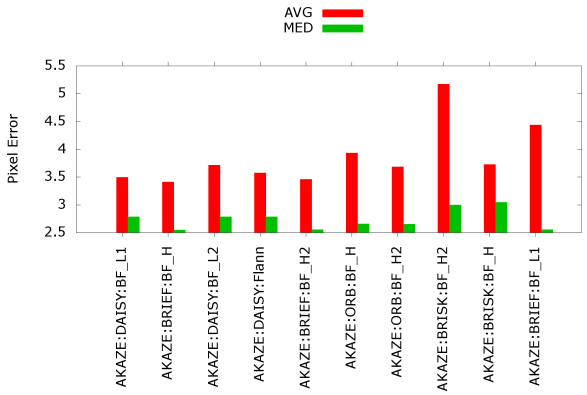

Figure 9. Average and median errors of top 10 methods for sequence 'Poster'.

In the test where the 'Flacon' object was used, we got similar results as in the case of 'Poster'. Both of the objects is rich in features, but the 'Flacon' is a spatial object. However, if we look at Fig. 10 where the methods with the lowest 10 median value were plotted, one can see that KAZE and SIFT had more feature points and can track these over more pictures than MSER or SURF after the fundamental filtering. Even though they had the lowest median values, the average errors of these methods were rather high.

However, if one takes a look at the methods with the lowest average error, then he/she can observe that AKAZE, KAZE and SURF present in the top 10. These methods can track more points then the previous ones and the median errors are just around 2.0 pixels.

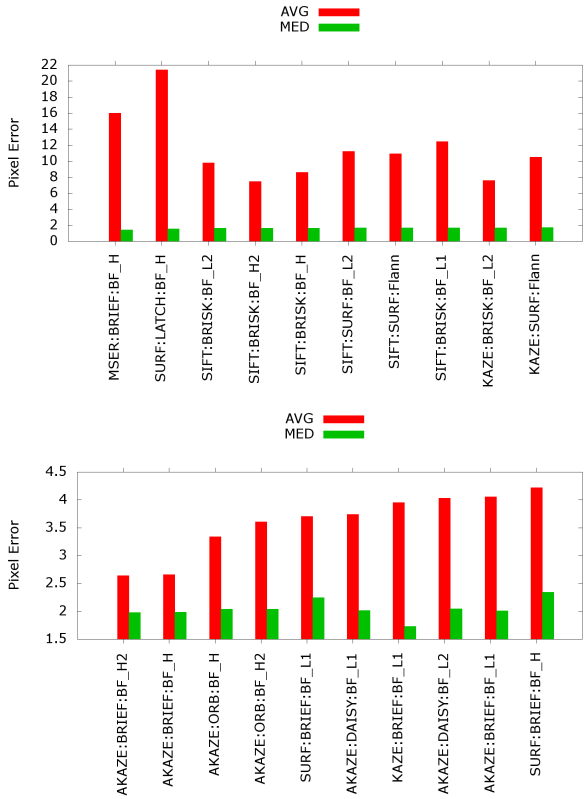

Figure 10. Top 10 method with the lowest median for sequence 'Flacon'. Chart are sorted by median (top) and average (bottom) values.

For the sequence 'Dinosaur' (Figure 11), the test object is very dark which makes feature detection hard. The number of available points is slightly more than 100. In this case, the overall winner of the methods is the SURF with both the lowest average and median errors. However, GFTT also present in the last chart too.

In the upper comparisons only the detectors were mentioned against each other. As one can see in

the charts, most of the methods used either DAISY, BRIEF, BRISK or SURF descriptors. From the perspective of matchers, it does not really matter which type of the matcher is used for the same detector descriptor type. However, if the descriptor gives a binary vector, then obviously the hamming distance outperforms the L2 or L1. But there are just slightly differences between the L1-L2 and H1-H2 distances.

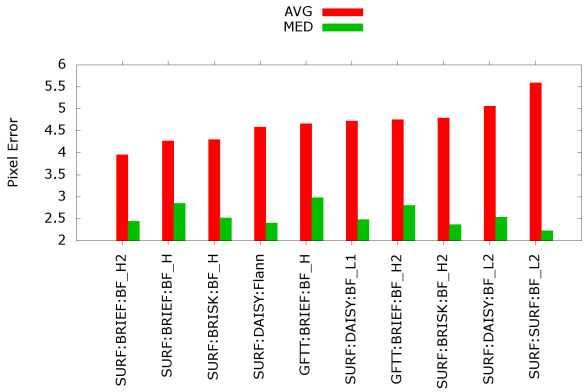

Figure 11. Top 10 methods (with lowest average error) on sequence 'Dinosaur'.

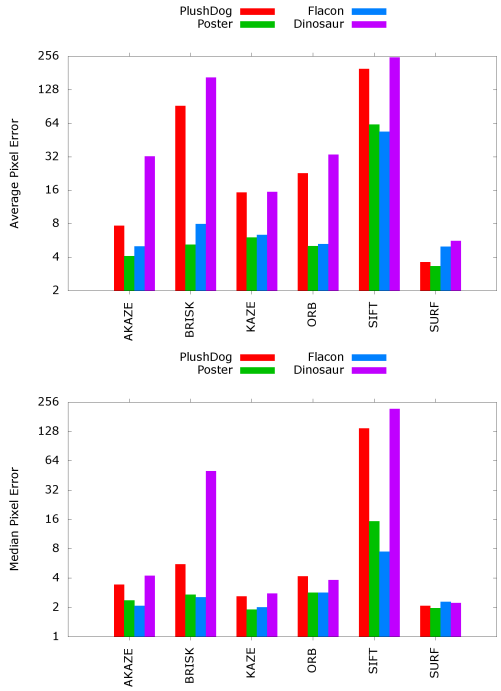

Figure 12. Overall average (top) and median (bottom) error values for all trackers and test sequences. The detectors and descriptors were the same.

Testing of algorithms with same detector and descriptor. In this comparison, only the detectors that have an own descriptor are tested. Always the best matchers is selected for which the error is minimal for the observed detector/descriptor.

As it can be seen in the log-scale charts in Fig. 12, the median error is almost the same for the AKAZE, KAZE, ORB and SURF trackers, but SURF is considered with the lowest average value. The tests 'Flacon' and 'Poster' result the lower pixel errors. On the other hand the rotation of the 'Dinosaur' was the hardest to track, it resulted much higher errors for all trackers comparing to the other tests.

5. Conclusions, Limitations, and Future Work

We quantitatively compared the well-known feature detectors, descriptors, and matchers implemented in OpenCV3 in this study. The GT datasets was generated by a structured-light scanner. The four testing objects were rotated by the turntable of our equipment. It seems to be clear that the most accurate feature for matching methods is the SURF [4] one proposed by Bay et al. It outperforms the other algorithms in all test cases. The other very accurate algorithms are KAZE [2]/AKAZE [17], they are the runner-up in our competition.

The most important conclusion for us is that such a comparison is a very hard task: for example, there are infinite number of possible error metrics; the quality is hardly influenced by the number of features, and so on. The main limitation here is that we can only test the methods in images of rotating objects. We are not sure that the same performance would be obtained if translating objects are observed. A possible solution to the extension of this paper is to compare the same methods on the Middlebury database and unify the obtained results for rotation and translation.

We hope that this paper is just the very first step of our research. We plan to generate more testing data, and more algorithms will also be involved into the tests. The GT dataset will be online, and an open-source testing system is also planned to be available soon [4].

References

[1] M. Agrawal and K. Konolige. Censure: Center surround extremas for realtime feature detection and matching. In *ECCV*, 2008. 3

[2] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. Kaze features. In *ECCV (6)*, pages 214–227, 2012. 2, 3, 8

[4] See http://web.eee.sztaki.hu

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

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008. 3, 8

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, pages 778–792, 2010. 3

[6] A. W. "Fitzgibbon, G. Cross, and A. Zisserman. "automatic 3D model construction for turn-table sequences". In *"3D Structure from Multiple Images of Large-Scale Environments, LNCS 1506"*, pages "155–170", "1998". 2

[7] P.-E. Forssn and D. G. Lowe. Shape descriptors for maximally stable extremal regions. In *ICCV*. IEEE, 2007. 6

[8] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94(3):335–360, 2011. 2

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 2

[10] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, 2011. 2, 3

[11] G. Levi and T. Hassner. LATCH: learned arrangements of three patch codes. *CoRR*, 2015. 3

[12] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, ICCV '99, pages 1150–1157, 1999. 3

[13] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings of the 11th European Conference on Computer Vision: Part II*, pages 183–196, 2010. 2

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC*, pages 36.1–36.10, 2002. 2

[15] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009. 4

[16] R. Ortiz. Freak: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517, 2012. 3

[17] A. B. Pablo Alcantarilla (Georgia Institute of Technolog), Jesus Nuevo (TrueVision Solutions AU). Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013. 2, 3, 8

[18] C. J. Pal, J. J. Weinman, L. C. Tran, and D. Scharstein. On learning conditional random fields for stereo - exploring model structures and approximate inference. *International Journal of Computer Vision*, 99(3):319–337, 2012. 1

[19] Z. Pusztai and L. Hajder. A Turntable-based Approach for Ground Truth Tracking Data Generation . In *VISAPP 2016*, pages 498–509, 2016. 2

[20] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *In Internation Conference on Computer Vision*, pages 1508–1515, 2005. 2

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, 2011. 2, 3

[22] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *Pattern Recognition - 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings*, pages 31–42, 2014. 1

[23] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47:7–42, 2002. 1

[24] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR (1)*, pages 195–202, 2003. 1

[25] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 519–528, 2006. 1

[26] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "a benchmark for the evaluation of rgb-d slam systems". In *"Proc. of the International Conference on Intelligent Robot Systems (IROS)"*, "2012". 2

[27] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE TRANS. PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 32(5), 2010. 3

[28] Tomasi, C. and Shi, J. Good Features to Track. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 593–600, 1994. 2