MatchBench: An Evaluation of Feature Matchers

JiaWang Bian\textsuperscript{1} Ruihan Yang\textsuperscript{2} Yun Liu\textsuperscript{2} Le Zhang\textsuperscript{3}  
Ming-Ming Cheng\textsuperscript{2} Ian Reid\textsuperscript{1} WenHai Wu\textsuperscript{4}  
\textsuperscript{1}University of Adelaide \textsuperscript{2}Nankai University \textsuperscript{3}Advanced Digital Sciences Center \textsuperscript{4}Huawei Technologies

Abstract. Feature matching is one of the most fundamental and active research areas in computer vision. A comprehensive evaluation of feature matchers is necessary, since it would advance both the development of this field and also high-level applications such as Structure-from-Motion or Visual SLAM. However, to the best of our knowledge, no previous work targets the evaluation of feature matchers while they only focus on evaluating feature detectors and descriptors. This leads to a critical absence in this field that there is no standard datasets and evaluation metrics to evaluate different feature matchers fairly. To this end, we present the first uniform feature matching benchmark to facilitate the evaluation of feature matchers. In the proposed benchmark, matchers are evaluated in three different aspects, involving matching ability, correspondence sufficiency, and efficiency. Also, their performances are investigated in different scenes and in different matching types. Subsequently, we carry out an extensive evaluation of different state-of-the-art matchers on the benchmark and make in-depth analyses based on the reported results. This can be used to design practical matching systems in real applications and also advocates the potential future research directions in the filed of feature matching.

1 Introduction

Feature matching is one of the most fundamental and active research areas in computer vision. The goal of matching is to build feature correspondences between different views of a scene or object. The correspondence search provides a basis for image based localization, tracking and reconstruction, so feature matchers are often used in many high-level applications such as Structure-from-Motion\textsuperscript{[1]} and Visual SLAM\textsuperscript{[2,3]}. Therefore, it is necessary to evaluate feature matchers, which would advance the development of both matching algorithms and related applications. However, to the best of our knowledge, no previous work targets the evaluation of feature matchers while they only focus on evaluating feature detectors\textsuperscript{[4,5]} and descriptors\textsuperscript{[5,6,7,8,9]}. This leads to a critical absence that there is no standard datasets and evaluation metrics to evaluate different feature matchers fairly.

To this end, we propose the first uniform feature matching benchmark to facilitate the analysis of feature matchers. In the proposed benchmark, matchers are evaluated in three different aspects, involving matching ability,
correspondence sufficiency, and efficiency. Here, the matching ability refers to how likely matchers perform correct matching between a pair of images, and the correspondence sufficiency refers to how many correspondences matchers proposed when they match an image pair correctly, and the efficiency refers to the speed of matching. They are all critical in high-level applications. For example, wrong matchings or inadequate correspondences would cause SfM/SLAM systems to function inappropriately, and slow matchings would cause high-level applications to be not able to work at real-time speed. In order to measure these different aspects, we propose two evaluation metrics, SP curves (with AUC score for showing an overall performance) and AP bars, which respond to the measurement of matching ability and correspondence sufficiency, respectively.

Instead of reinventing the wheel, our benchmark dataset is constructed by collecting image sequences from existing SfM/SLAM datasets [10,11,12]. This is because a) one goal of this paper is to advance the development of SfM/SLAM [13] by improving the matching techniques, and thus performing the evaluation on SfM/SLAM datasets is the most straightforward; b) existing SfM/SLAM datasets [10,11,12] are large enough and they cover a wide range of scenes, involving indoor offices, different objects, outdoor street views, and urban buildings. On the other hand, although images are off-the-shelf, we contribute by selecting and re-organizing them for enabling both short-baseline and wide-baseline feature matching evaluation, responding to matching problems in Visual SLAM and Structure-from-Motion, respectively. What’s more, we make our dataset extensible by providing easy-to-use tools for re-organizing some popular SLAM/SfM datasets at our format. It enables researchers to run our evaluation protocol on their own dataset for choosing proper matchers that meet their requirements.

Subsequently, we carry out a comprehensive evaluation of different state-of-the-art feature matchers [13,14,15,16,17,18,19,20,21] on the proposed benchmark, and then conduct in-depth analyses based on the results. This can be used to design practical matching systems in real applications and also advocates the potential future research directions in the filed of feature matching.

The contributions of this paper are as following:

- a) we propose the first uniform feature matching benchmark to facilitate the evaluation of feature matchers in different scenes and in different aspects, which enables researchers to develop and evaluate their new algorithms more conveniently.

- b) we carry out an extensive evaluation of various state-of-the-art matchers and make in-depth analyses, which encourage researchers to design better practical matchers in real applications and also advocate the potential future research directions in the filed of feature matching.

On the other hand, the novelty of this paper involves proposing three different aspects for evaluating matchers, designing two evaluation metrics for facilitate the analysis of matching ability and correspondence sufficiency, and constructing
MatchBench: An Evaluation of Feature Matchers

(re-organizing) benchmark datasets for enabling both short-baseline and wide-baseline feature matching evaluation.

We organize the paper by giving an overview of feature matchers in Sec. 2, introducing evaluation metrics in Sec. 3, constructing benchmark datasets in Sec. 4, and evaluating feature matchers in Sec. 5. Finally, some discussions of this work are listed in Sec. 6, and conclusions are given in Sec. 7.

2 Feature matchers overview

A typical feature matcher proceeds by extracting local features \cite{13,14,15}, and followed by matching features by using a nearest-neighbor approach \cite{22}, and finally selecting good correspondences \cite{16,20,18} from the tentative correspondence set. The selected correspondences would be fed into a RANSAC framework for fitting a global geometry model \cite{28} from them, and the outliers are further rejected by using the estimated model. Often, the estimated geometry model as well as final correspondences are delivered in Structure-from-Motion \cite{1} and Visual SLAM systems \cite{3}. We give an overview of feature matchers below.

SIFT matcher \cite{13} is a standard method in this field. It follows the typical pipeline we mentioned above, where FLANN \cite{22} is often used to perform fast (approximated) nearest-neighbor matching, and RATIO \cite{13} is used to select good correspondences which compares the lowest feature distance and the second lowest feature distance for recognizing good ones. SIFT matcher is widely used in different applications, and we regard it to be the baseline for analyzing other matchers.

There are two main research directions for boosting matchers’ performances or efficiencies, including designing better local features \cite{14,15,29,30,31,32,33} and better matching solutions \cite{16,19,18,17,20}. Here, local features are reviewed and evaluated in many previous works \cite{6,5,7,8,9}, but matching solutions are few discussed. Therefore, we focus on introducing the latter below.

Graph matchers \cite{34,35,36,17} search for geometry consistent correspondences between two sets of features, rather than performing nearest-neighbor matching and selecting good correspondences like a typical matching system. They optimize a global consistency score and can cope with higher-order constraints (involving more than one match). However, they are not well suited for a high outlier rate, and their time and space complexity grows exponentially with the order, which limits in real applications to a few hundred feature points.

KVLD matcher \cite{16} proposes a virtual Line descriptor and a semi-local matching method based on this descriptor for correspondence selection. It makes good use of constraints in both photometry and geometry, and correspondences that pass the verification in both domains are recognized to be good. The methods works well in strong-texture scenes while suffers in weak-texture scenes because in this scenario photometry-based solutions may function inappropriately.

CODE matcher \cite{18} proposes an optimization based approach for finding a globally smooth correspondence set. Employing powerful ASIFT \cite{37} feature, it
performs ultra-robust wide-baseline matching and proposes sufficient correspondences. Based on CODE, **RepMatch matcher** [19] proposes a geometry-aware approach to tackle the challenges of repetitive structures. It improves the performance again but introduces a higher complexity at the same time. However, these two matchers [19][18] have huge computational costs, although they are very powerful.

**GMS matcher** [20] proposes a correspondence selection method called **grid-based motion statistics**. It is fast and robust to recognize good correspondences. Adopting cheap and rich ORB [15] feature, the whole matcher can perform high-quality matching while achieving real-time performances.

Finally, with respect to the **number of correspondences**, matchers can also be divided into two classes, **sparse feature matchers**, and **rich feature matchers**. Here, CODE [18], RepMatch [19], and GMS [20] fall into the group of **rich feature matchers**, since their output correspondences are much more than other **sparse matchers**. They are all recently proposed and show high-quality feature matching. We compare these two classes of matchers in our evaluation.

### 3 Evaluation metrics

The inputs of a matcher are two images and the outputs are correspondences between them. It sounds straightforward to benchmark the output correspondences, but one may **find that it is impractical** due to the difficulty of generating the highly-quality (accurate and dense) ground truth. To our knowledge, there are two methods for ground-truth correspondences generation: a) The first approach is projecting a pixel in one image to other images by using Homography (see details in [35]). However, this can only be used in a planar scene **but not be applicable in a generic non-planar scene**. b) The other approach enables projection in non-planar scenes by using internal camera parameters (calibration matrix), external camera parameters (camera poses), and depth images. However, the method turns out to be **lacking of density and precision** due to the low-quality (sparse and low-precision) depth, leading to less conclusive results.

To this end, we propose to feed correspondences into a pose estimator and benchmark the results of pose estimation, instead of directly evaluating correspondences. In this design, the **pose error** (compared with the ground-truth pose) implies how well an image pair is matched, and a matched pair would be **regard to be correct** if its pose error is less than certain threshold. For estimating the relative camera pose $T$, we firstly estimate the essential matrix $E$ from correspondences $C$ and internal camera parameters (calibration matrix) $K$:

$$E \leftarrow C, K \quad (1)$$

Alternatively, we can also estimate the fundamental matrix $F$ from correspondences $C$ and convert it to $E$ given $K$:

$$F \leftarrow C \quad (2)$$
$E = K^{-1} \cdot F \cdot K$  (3)

Then we get:

$$T \leftarrow \{C, E, K\}$$  (4)

More details about pose estimation can be referenced in [28, 38].

In order to measure the correctness of a matched pair, we split $T$ into a rotation matrix $R$ and a translation vector $t$, and then compare them with the ground-truth $R_{gt}$ and $t_{gt}$. This leads to a rotational error $e_r$ and translational error $e_t$. Here, both errors are in degrees. Specifically, $e_r$ is computed from the transformation matrix from $R_{gt}$ to $R$ as did in KITTI [10], and $e_t$ is the angle between vectors $t_{gt}$ and $t$. Note that two translational vectors are in different scales because the scale cannot be estimated given monocular image pairs (see details in [38]). We set the camera pose error to be:

$$e = \max(e_r, e_t)$$  (5)

Then an image pair would be recognized as a correct match if its pose error $e$ is less than certain threshold.

Given the above method to verify a matched pair, we further propose two metrics for benchmarking a matcher: SP (Success ratio / Pose error threshold) curves and AP (Averaged number of correspondences / Pose error threshold) bars. SP curves show the change of success ratio, the percentage of correctly matched pairs to all image pairs, with increasing pose error thresholds. This responds to matching ability measurement. AP bars illustrate the mean number of correspondences averaged over correctly matched pairs (the threshold of 5 degrees is used here). It measures the correspondence sufficiency of matchers.

Besides, for showing an overall matching ability of matchers, we propose to compute the AUC score (Area Under Curve) of SP curves. As the pose error thresholds are discrete, the AUC score of matchers equals to the mean value of its success ratios on all pose error thresholds.

### 4 Benchmark dataset

One goal of this paper is to evaluate feature matchers for advancing the development of Visual SLAM/SfM [1, 2, 3]. Therefore, rather than reinventing the wheel, we construct our benchmark dataset by collecting image sequences from existing SLAM/SfM datasets, including TUM SLAM dataset [11], KITTI odometry benchmark [10], and Strecha SfM dataset [12]. They not only provide real-world image sequences of different scenes, but also provide the precise camera trajectory (camera positions) as the ground truth. Besides, we split the dataset into two portions, involving short-baseline matching and wide-baseline matching portions. In different portions, the methods to construct image pairs are different. We introduce the dataset below.

**Dataset description.** Our dataset contains eight image sequences with the first four sequences for short-baseline matching evaluation and the last four sequences for wide-baseline matching evaluation. They are selected from three
datasets, including TUM dataset \cite{11} where videos of indoor scenes are captured at 30\textit{fps} and the sensor resolution is 640 \times 480, KITTI dataset \cite{10} where video sequences of street views are captured at 10\textit{fps} and the image resolution is 376 \times 1241, and Strecha dataset \cite{12} where authors provide image sequences of urban buildings and the resolution is 3072 \times 2048. The screen-shot and description of selected image sequences are illustrated in Fig. 1 and Tab. 1, respectively. Here, sequence 04 is from KITTI dataset \cite{10} and sequence 05 is from Strecha dataset \cite{12}. They are easier than other sequences, since others are from TUM dataset \cite{11} where the texture of scenes is weaker and the image resolution is lower. Especially, sequence 02(07) and sequence 03(08) are challenging, as the former captures a non-planar object and the latter captures a low-texture object.

Table 1. The description of the selected image sequences.

| Type | Sequences       | Images Pairs | Resolution       | Property       |
|------|-----------------|--------------|------------------|----------------|
| Short| 01-office       | 2583         | 480 \times 640   | indoor, office |
|      | 02-teddy       | 2405         | 480 \times 640   | indoor, non-planar |
|      | 03-large-cabinet | 1006         | 480 \times 640   | indoor, weak texture |
|      | 04-kitti       | 4542         | 376 \times 1241  | outdoor, street-view |
| Wide | 05-castle      | 30           | 3072 \times 2048 | outdoor, urban |
|      | 06-office-wide | 173          | 480 \times 640   | same with 01   |
|      | 07-teddy-wide  | 161          | 480 \times 640   | same with 02   |
|      | 08-large-cabinet-wide | 68 | 480 \times 640 | same with 03 |

Fig. 1. The screen-shots of the selected images sequences. The sequences 01-03 (06-08), 04, 05 are collected from TUM dataset \cite{11}, KITTI dataset \cite{10}, and Strecha dataset \cite{12}, respectively.
**Short-baseline matching portion.** Three sequences (01-03) are from TUM dataset \[11\] and one sequence (04) is from KITTI dataset \[10\]. Every video sequence is divided into \(m\) non-overlapping fragments and each fragment contains \(k\) frames. In each fragment, the first frame is set to be the reference image and other \(k-1\) frames will be matched to it. Here, \(k\) is set to be 15 for sequences 01-03 and be 5 for sequence 04, since they are captured at 30 fps and 10 fps, respectively. It means the time length of each fragment is 0.5 seconds. This results in \(n-m\) image pairs in each sequence, where \(n\) is the number of images in the sequence.

**Wide-baseline matching portion.** The fifth sequence (05-castle) is selected from Strecha \[12\] dataset, and other sequences 06-08 are sub-sampled from sequences 01-03. For the sequence 05, we run all possible \(n \times (n-1)/2\) pairs, where \(n\) is the number of images in the sequence. For sequences 01-03, we extract the first image of every fragment (where each fragment contains 15 frames) in each sequence, leading to sequences 06-08. Then for each sequence, every image is matched to the next at most 9 images. Since the the sensor is 30 fps, every image is matched to the next frames captured within 5 seconds. This is based on our observation that most pairs beyond 5 seconds are with no overlap. Note that in this portion not all pairs are with overlap, but this doesn’t influence the relative performance of different matchers because false pairs are nearly impossible to be ”matched and estimated correctly” by any matcher.

5 Experiments

We perform exhaustive evaluation of different feature matchers in this section. As described in Sec. 1, matchers are evaluated in terms of matching ability, correspondence sufficiency, and efficiency. They are also evaluated in different type of matching tasks, involving short-baseline matching and wide-baseline matching. Evaluation settings, experimental results, and analyses are given in the following sections.

5.1 Evaluation setting

**Evaluated matchers.** For a comprehensive evaluation, we collect various state-of-the-art matchers. They fall into two main categories, distinctive local features and powerful matching solutions, as described in Sec. 2. The first category includes SIFT \[13\], SURF \[14\], ORB \[15\], BRISK \[31\], KAZE \[29\], AKAZE \[30\], DLCO \[33\], FREAK \[32\], BinBoost \[39\], LATCH \[40\], and DAISY \[41\] total 11 methods. Here, the last five methods (DLCO, FREAK, BinBoost, LATCH, DAISY) only provide feature descriptors and no detector is available. Therefore, we concatenate them (except for FREAK \[32\]) with the SIFT \[13\] detector, and combine FREAK descriptor with SURF \[14\] detector as suggested in OpenCV samples. These features follow the classical matching pipeline that features are matched by using a nearest-neighbor approach and correspondences are selected by using RATIO (the threshold is 0.8 as widely used in applications). The
second category includes KVLD \cite{16}, GAIM \cite{17}, CODE \cite{18}. RepMatch \cite{19}, and GMS \cite{20} matchers. A brief description of different matchers can be seen in Sec. 2.

The problem associated short-baseline matching involves video-based applications such as Visual SLAM \cite{2,3} where the efficiency is quite critical. It would be less meaningful to evaluate a slow matcher if it could not be integrated into real-time applications. Therefore, we exclude slow matchers (KVLD, GAIM, CODE, and RepMatch) in short-baseline matching portion, as they seem far away from enabling fast matching even though GPU is available. For wide-baseline matching, all matchers are evaluated.

Camera pose estimation. We adopt two pose estimators for camera pose estimation. The first one is from OpenCV library which implements five-points \cite{28} method for essential matrix estimation within a robust RANSAC \cite{23} framework. The estimator is well-tuned and widely used for estimating relative camera pose from a set of correspondences. However, we empirically find that this estimator doesn’t work well for rich feature matchers (CODE \cite{18}, RepMatch \cite{19}, GMS \cite{20}), as their output correspondences are much more than traditional sparse matchers. Therefore, we propose to use the pose estimator built in RepMatch \cite{19} for these three rich matchers. We also try to use this estimator with other sparse matchers, like SIFT \cite{13}. However, the results show that the OpenCV estimator is consistently better. Therefore, for sparse matchers, we still use the OpenCV estimator.

Implementation details. The implementation of all local features is from OpenCV library. We use their default parameters for extracting features, except for ORB \cite{15} feature. Here, the default $nfeatures$ of ORB implementation is 500 which limits the maximum number of detected features. We manually assign a big number (100,000) to it for breaking this limitation. Note that the number of detected features are often much more lower than this value in practice. On the other hand, in order to match features, we adopt FLANN matcher \cite{22} with Euclidean distance for real-value features (SIFT, SURF, KAZE, DLCO, DAISY) and adopt Brute-force matcher with Hamming distance for matching binary features (ORB, BRISK, AKAZE, FREAK, BinBoost, LATCH) for the best trade-off between performances and efficiency. This is a widely used setting in feature matching. What’s more, for others matchers, we follow the default setting provided by authors. Specifically, KVLD \cite{16} adopts SIFT feature; CODE \cite{18} and RepMatch \cite{19} employ ASIFT \cite{37} feature; GAIM \cite{17} simulates images and extracts SURF \cite{14} feature; GMS \cite{20} adopts ORB \cite{15} feature (extracting at most $10K$ interest points).

Speed testing. For comparing the time consumption of different matchers fairly, we run all algorithms in one computer where CPU is Intel i7-6700K and GPU is NVIDIA GTX 1080. The first 200 image pairs in sequence 01 are used to evaluate the speed of matchers, and the averaged time consumption of matchers is reported in Tab. 3. Note that most feature detection and nearest-neighbor matching methods can be accelerated by GPU while correspondence selection
approaches are not trivial to be accelerated. Therefore, the latter may be the bottleneck in real applications.

5.2 Evaluation results and analyses

Table 2. The AUC score of matchers. The best three methods in each sequence are labeled in red, green, and blue colors, respectively. Besides, matchers which outperform the baseline (SIFT matcher) are labeled in bold font.

| Matchers | short-baseline portion | wide-baseline portion |
|----------|------------------------|-----------------------|
|          | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |
| SIFT     | 0.375 | 0.415 | 0.137 | 0.939 | 0.396 | 0.538 | 0.179 | 0.198 |
| SURF     | 0.360 | 0.449 | 0.159 | 0.919 | 0.385 | 0.456 | 0.180 | 0.180 |
| ORB      | 0.265 | 0.328 | 0.078 | 0.874 | 0.338 | 0.480 | 0.141 | 0.115 |
| BRISK    | 0.288 | 0.260 | 0.098 | 0.897 | 0.329 | 0.468 | 0.110 | 0.127 |
| KAZE     | 0.382 | 0.408 | 0.167 | 0.908 | 0.377 | 0.557 | 0.215 | 0.177 |
| ACKZE    | 0.340 | 0.388 | 0.121 | 0.897 | 0.358 | 0.456 | 0.169 | 0.127 |
| DLCO     | **0.379** | **0.420** | 0.131 | 0.935 | 0.415 | 0.543 | 0.195 | 0.210 |
| FREAK    | 0.339 | 0.370 | 0.128 | 0.927 | 0.348 | 0.435 | 0.134 | 0.156 |
| BinBoost | **0.378** | **0.404** | 0.134 | 0.936 | 0.375 | 0.508 | 0.170 | 0.185 |
| LATCH    | 0.348 | 0.392 | 0.124 | 0.909 | 0.349 | 0.389 | 0.134 | 0.138 |
| DAISY    | 0.354 | 0.388 | 0.112 | 0.925 | 0.356 | 0.493 | 0.171 | 0.169 |
| KVLD     | / | / | / | / | **0.403** | **0.492** | **0.150** | **0.142** |
| GAIM     | / | / | / | / | 0.210 | 0.360 | 0.142 | 0.100 |
| CODE     | / | / | / | / | **0.447** | **0.809** | **0.436** | **0.476** |
| RepMatch | / | / | / | / | / | **0.535** | **0.766** | **0.425** | **0.473** |
| GMS      | **0.508** | **0.605** | **0.251** | **0.955** | 0.433 | **0.605** | **0.332** | **0.347** |

The experimental results of short-baseline matching and wide-baseline matching are illustrated in Fig. 2 and Fig. 3, respectively. The AUC score of matchers is shown in Tab. 2 and the time consumption of matchers is shown in Tab. 3. These results enable us to analyze the matching ability, correspondence sufficiency, as well as efficiency of different matchers. We make the following analyses.

a) the experimental data (image size, scene type, and etc) influences the performance of matchers significantly. Seeing Fig. [2] or Tab. [2], one can find that the matching abilities of matchers are high in sequence 04, and they are significantly lower in sequences 01-03. At the same time, the performance gap of different matchers is narrow in sequence 04 and it is wide in sequences 01-03. Due to this, we may regard matching ability to be the most vital factor to choose good matchers in sequences 01-03, but we may pay more attention on the efficiency or correspondence sufficiency of matchers in sequence 04 for the best trade-off. Therefore, we suggest researchers re-organizing their own dataset and running our evaluation protocol on it for selecting appropriate matchers before developing an real application.
**Fig. 2.** Evaluation results on the short-baseline matching portion. SP curves show the success ratio of matchers with changing pose error thresholds and AP bars show the number of correspondences which is averaged over correctly matched pairs.

b) **rich feature matchers vs sparse feature matchers.** Three matchers (CODE [18], RepMatch [19], and GMS [20]) fall into the first class while other matchers fall into the second class. First, in terms of matching ability, rich matchers outperform the sparse matchers. This is demonstrated in Tab. 2 where GMS
matcher [18] outperforms other (sparse) matchers consistently in short-baseline portion and rich matchers (CODE [18], RepMatch [19], and GMS [20]) outperform others in wide-baseline portion (except for the case that KVLD matcher [16] slightly outperforms CODE and GMS in sequence 05). Second, with respect
to correspondence sufficiency (see Fig. 2 or Fig. 3), rich matchers naturally outperform sparse matchers. Third, with regard to efficiency, CODE [18] and RepMatch [19] are much more slower than most sparse matchers (except for GAIM [17]) even though GPU is adopted, but GMS [20] can show real-time performances.

c) local feature extractors. We regard SIFT feature [13] to be the baseline for analyzing other local features. First, with regard to matching ability (see Tab. 2), three features (SURF [13], KAZE [29], DLCO [33]) show equivalent and higher performances than SIFT feature, while other features are not as good as that. Second, in terms of correspondence sufficiency (see Fig. 2 or Fig. 3), ORB feature [15] obviously outperforms the baseline and other features. Third, with respect to efficiency (see Tab. 3), four binary features (ORB [15], AKAZE [30], BRISK [31], FREAK [32]) outperform the baseline (SIFT [13]).

d) matching solutions. As before, We regard SIFT matcher [13] to be the baseline. First, with regard to matching ability (see Tab. 2), three rich feature matchers (CODE [18], RepMatch [19], and GMS [20]) outperform the baseline consistently. KVLD matcher [10] beats the baseline in sequence 05 and is beaten by the latter in other sequences. GAIM matcher [17] shows consistently lower performances than the baseline. Second, with regard to correspondence sufficiency (see Fig. 2 or Fig. 3), three rich feature matchers (CODE [18], RepMatch [19], and GMS [20]) outperform the baseline, and other matchers (KVLD and GAIM) show similar performances with the baseline. Third, with respect to efficiency (see Tab. 3), only GMS matcher shows higher speed than the baseline by adopting GPU acceleration, and other matchers are much more slower than the baseline.

e) The best generic matcher. GMS matcher [20] outperforms sparse matchers in terms of matching ability and correspondence sufficiency, although it is weaker than other two rich matchers (CODE [18], RepMatch [19]). With respect to efficiency, it is several orders of magnitude faster than rich feature matchers (CODE and RepMatch), and is efficient enough to enable real-time performances by using GPU. Therefore, we get the conclusion that GMS matcher [20] shows the best trade-off among matching ability, correspondence sufficiency, and efficiency.

6 Discussion

Our primary goal is to set up an uniform benchmark to evaluate feature matchers. We have made significant efforts on making it reasonable and convenient to use as well as possible. The proposed benchmark is discussed below.

Contribution and novelty. As introduced in Sec. 1, the contribution of this paper includes i) we set up the first uniform feature matching benchmark to facilitate the evaluation of feature matchers, which enables researchers explore and develop their matchers conveniently. ii) we conduct exhaustive evaluation of different state-of-the-art matchers, where the results and conclusions can be used to design practical matchers in real applications and also advocate
Table 3. The time consumption of different matchers. Values in brackets mean GPU time, and others stand for CPU time.

| Matchers | Feature numbers | Detection time (ms) | Matching time (ms) | Selection time (ms) |
|----------|-----------------|---------------------|-------------------|---------------------|
| SIFT     | 1082            | 56.4                | 21.2              |                     |
| SURF     | 1432            | 63.0                | 19.7              |                     |
| ORB      | 3359            | 10.3                | 38.2              |                     |
| AKAZE    | 726             | 29.7                | 3.6               |                     |
| BRISK    | 1100            | 17.8                | 5.4               |                     |
| KAZE     | 1060            | 157.0               | 15.5              |                     |
| DLCO     | 1082            | 430.4               | 21.7              |                     |
| FREAK    | 919             | 43.9                | 3.2               |                     |
| BinBoost | 1082            | 94.0                | 4.2               |                     |
| LATCH    | 984             | 86.1                | 3.3               |                     |
| DAISY    | 1082            | 79.3                | 25.2              |                     |
| KVLD     | 1082            | 56.4                | 21.2              | 540.1               |
| GAIM     | 45345           | 6783.2              | 1550.6            | 7145.1              |
| CODE     | 64609           | (1365.0)            | (970.1)           | 3079.6              |
| RepMatch | 9463            | 33.0                | (12.4)            | 1079.6              |
| GMS      | 9463            |                     |                   | 1.3                 |

the potential future research directions in the field of local feature extraction and matching solutions. On the other hand, the novelty involves proposing three different aspects to evaluate matchers, designing corresponding evaluation metrics, and creating (re-organizing) benchmark datasets for enabling both short-baseline and wide-baseline feature matching evaluation.

**Evaluation metrics.** The proposed SP curves (with AUC score) and AP bars rely on camera pose estimation which we use to judge whether a pair is matched correctly. Therefore, the performance of matching is not only leaded by feature matchers but also pose estimators. One may concern that pose estimators could not work perfectly and it would lead to an incorrect comparison of matchers. For example, estimators may sometimes fail to get a correct pose estimation even though an image pair is matched well. However, we argue that the current solution is reasonable because two-view pose estimation is an essential part in SfM/Monocular SLAM where the estimated camera pose is directly used to initialize the system even though other pairs’ poses could be refined in further processing when the system has been initialized. Therefore, our current evaluation implies how likely a matcher can enable correct initialization in SfM or Monocular SLAM. This is a very practical and vital problem!

**Benchmark datasets.** Although the benchmark dataset covers a wide range of scenes, one may concern that the images in wide-baseline portion are not as diverse as images in some SfM datasets, like Internet-image collections [42] where images are captured from many different cameras. We exclude these diverse datasets because they often cannot provide precise ground-truth camera positions for evaluation. One possible solution to use these dataset for evaluation is reconstructing 3D models using SfM tools and regarding their estimated camera positions as ”ground-truth”. However, we argue that it is not reliable
enough and instead propose to sub-sample video sequences with precise camera positions for our evaluation. Besides, even though the current single-camera setting is not as diverse as Internet-image datasets, it is still practical in many real-life scenarios. For example, we sometimes may need to reconstruct 3D models of an office (or a living room) from unordered photos captured by a smart phone. Finally, we will still be considering how to introduce more diversified datasets while keeping the ground truth accurate.

**Evaluated methods.** The proposed benchmark not only can be used to benchmark feature matchers but also pose estimators. Since currently we are more interested in feature matchers, various pose estimators are not introduced in our evaluation. In order to maximize matchers’ performances, we have adopted two state-of-the-art pose estimators and select the properest possible one for each matcher. Limited to the page length, we would explore more pose estimators and add more ablation studies in the future work.

### 7 Conclusions

This paper proposes the first uniform benchmark to evaluate feature matchers. It suggests analyzing matchers in three different aspects, including matching ability, correspondence sufficiency, and efficiency. In order to measure these different properties, the paper presents two novel evaluation metrics. On the other hand, the proposed benchmark dataset covers a wide range of scenes and can be used to evaluate matchers in different type of problems, involving short-baseline matching and wide-baseline matching. What’s more, comprehensive evaluation of different feature matchers is carried out and results are useful for researchers to design practical matching systems in real applications.

### References

1. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 4104–4113
2. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: Real-time single camera slam. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) 29 (2007) 1052–1067
3. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. IEEE Transactions on Robotics (TOR) 31 (2015) 1147–1163
4. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A comparison of affine region detectors. International Journal on Computer Vision (IJCV) 65 (2005) 43–72
5. Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3d objects. International Journal on Computer Vision (IJCV) 73 (2007) 263–284
6. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) 27 (2005) 1615–1630
7. Heinly, J., Dunn, E., Frahm, J.M.: Comparative evaluation of binary features. In: European Conference on Computer Vision (ECCV). Springer (2012) 759–773
8. Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K.: HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2017) 5173–5182
9. Schönberger, J.L., Hardmeier, H., Sattler, T., Pollefeys, M.: Comparative evaluation of hand-crafted and learned local features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2017) 6959–6968
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2012) 3354–3361
11. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: IEEE International Conference on Intelligent Robots and Systems (IROS). (2012)
12. Strecha, C., Von Hansen, W., Van Gool, L., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2008) 1–8
13. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal on Computer Vision (IJCV) 60 (2004) 91–110
14. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding (CVIU) 110 (2008) 346–359
15. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: IEEE International Conference on Computer Vision (ICCV), IEEE (2011) 2564–2571
16. Liu, Z., Marlet, R.: Virtual line descriptor and semi-local matching method for reliable feature correspondence. In: British Machine Vision Conference (BMVC). (2012) 16–1
17. Collins, T., Mesejo, P., Bartoli, A.: An analysis of errors in graph-based keypoint matching and proposed solutions. In: European Conference on Computer Vision (ECCV). Springer (2014) 138–153
18. Lin, W.Y., Wang, F., Cheng, M.M., Yeung, S.K., Torr, P.H., Do, M.N., Lu, J.: Code: Coherence based decision boundaries for feature correspondence. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) (2017)
19. Lin, W.Y., Liu, S., Jiang, N., Do, M.N., Tan, P., Lu, J.: Repmatch: Robust feature matching and pose for reconstructing modern cities. In: European Conference on Computer Vision (ECCV), Springer (2016) 562–579
20. Bian, J., Lin, W.Y., Matsushita, Y., Yeung, S.K., Nguyen, T.D., Cheng, M.M.: GMS: Grid-based motion statistics for fast, ultra-robust feature correspondence. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2017) 4181–4190
21. Schönberger, J.L., Price, T., Sattler, T., Frahm, J.M., Pollefeys, M.: A vote-and-verify strategy for fast spatial verification in image retrieval. In: Asian Conference on Computer Vision, Springer (2016) 321–337
22. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1) 2 (2009) 2
23. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24 (1981) 381–395
24. Chum, O., Matas, J.: Matching with prosac-progressive sample consensus. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1., IEEE (2005) 220–226
25. Torr, P.H., Zisserman, A.: Mlesac: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding (CVIU) 78 (2000) 138–156
26. Rousseeuw, P.J., Leroy, A.M.: Robust regression and outlier detection. Volume 589. John wiley & sons (2005)
27. Raguram, R., Chum, O., Pollefeys, M., Matas, J., Frahm, J.M.: Usac: a universal framework for random sample consensus. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) 35 (2013) 2022–2038
28. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) 26 (2004) 756–770
29. Alcantarilla, P.F., Bartoli, A., Davison, A.J.: Kaze features. In: European Conference on Computer Vision (ECCV), Springer (2012) 214–227
30. Alcantarilla, P.F., Solutions, T.: Fast explicit diffusion for accelerated features in nonlinear scale spaces. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) 34 (2011) 1281–1298
31. Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: Binary robust invariant scalable keypoints. In: IEEE International Conference on Computer Vision (ICCV), IEEE (2011) 2548–2555
32. Alahi, A., Ortiz, R., Vanderghynst, P.: Freak: Fast retina keypoint. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2012) 510–517
33. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) 36 (2014) 1573–1585
34. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: IEEE International Conference on Computer Vision (ICCV). Volume 2., IEEE (2005) 1482–1489
35. Zhou, F., De la Torre, F.: Deformable graph matching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2013) 2922–2929
36. Zhou, F., De la Torre, F.: Factorized graph matching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2012) 127–134
37. Morel, J.M., Yu, G.: Asift: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences 2 (2009) 438–469
38. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
39. Trzcinski, T., Christoudias, M., Fua, P., Lepetit, V.: Boosting binary keypoint descriptors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2013) 2874–2881
40. Levi, G., Hassner, T.: Latch: learned arrangements of three patch codes. In: Applications of Computer Vision (WACV), IEEE (2016) 1–9
41. Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide-baseline stereo. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) 32 (2010) 815–830
42. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building rome in a day. Communications of the ACM 54 (2011) 105–112