Towards a Fair Evaluation of Feature Extraction Algorithms Robustness in Structure from Motion

Dina M. Taha[1], Hala H. Zayed[2], Shady Y. El-Mashad[3]
Faculty of Engineering (at Shoubra), Benha University, Cairo, Egypt[1, 3]
Faculty of Computers and Artificial Intelligence, Benha University, Benha, Egypt[2]
School of Information Technology and Computer Science, Nile University, Cairo, Egypt[2]

***Abstract*—Structure from Motion is a pipeline for 3D reconstruction in which the true geometry of an object or a scene is inferred from a sequence of 2D images. As feature extraction is usually the first phase in the pipeline, the reconstruction quality depends on the accuracy of the feature extraction algorithm. Fairly evaluating the robustness of feature extraction algorithms in the absence of reconstruction ground truth is challenging due to the considerable number of parameters that affect the algorithms' sensitivity and the tradeoff between reconstruction size and error. The evaluation methodology proposed in this paper is based on two elements. The first is using constrained 3D reconstruction, in which only fixed numbers of extracted and matched features are passed to subsequent phases. The second is comparing the 3D reconstructions using size-error curves (introduced in this paper) rather than the value of reconstruction size, error, or both. The experimental results show that the proposed methodology is more transparent.**

Keywords—Feature extraction; feature matching; structure from motion; 3D reconstruction

I. Introduction

Structure from Motion (SfM) is a processing pipeline intended for reconstructing 3D models of objects from a sequence of 2D views (images). It involves several different phases in which various algorithms can be used, such as feature detection, feature description, feature matching (correspondence), triangulation, and finally 3D reconstruction. In addition, pruning or masking-out outliers may be performed between different phases. As it commonly starts with detecting features and computing descriptors, its accuracy depends mainly on the robustness of the used feature extraction algorithm. This raises a critical question; which of the available feature detectors-descriptors is best for SfM [1], [2]?

Feature detection and description is a low-level process that can be considered the cornerstone and is usually used as the starting point of many applications in computer vision. A feature (keypoint) is an interesting or important piece of information, such as points, blobs, corners, edges, junctions, etc., that is relevant to the solution of the computational task (correspondence) behind a specific application (3D reconstruction). Accordingly, a feature detector is an algorithm that can detect these interesting keypoints in a given image. On the other hand, a feature descriptor describes each detected feature in an image by assigning it a distinctive identity formed by the pixels within a certain neighborhood. This identity enables effective recognition of the corresponding features during matching [3], [4]. After feature detection, detected features can be pruned using either a threshold for the detector response or by selecting the features with the highest response.

The simplest strategy for matching two sets of feature descriptors and finding the corresponding features is brute-force matching. In this strategy, each feature descriptor in the first set is compared with each feature descriptor in the second set. This allows for finding all possible pairs of corresponding features. However, it requires extensive computational resources. A better and faster strategy for feature matching, especially in large datasets, is to adopt a hash table or a multi-dimensional search tree as an indexing data structure for rapidly retrieving neighbors (features) from the second set that are nearest to a given feature from the first set [5].

The list of corresponding features can be pruned by comparing the similarity (distance) of each pair with a predefined threshold representing the maximum allowed distance. Only pairs that satisfy the predefined threshold are kept, while other pairs are discarded. The downside of this method is the difficulty of deciding the appropriate threshold. In addition, applying the same threshold to all pairs can be inaccurate in some circumstances, as the distance of a correct match can be greater than that of an incorrect match. A better pruning strategy in such cases is to select only the best pair (the one with the smallest distance) for every feature or adopt the Nearest Neighbor Distance Ratio (NNDR) [6]. Matching results can be further pruned by selecting the pairs with the smallest distance.

The remaining incorrect matches (outliers) can be removed or filtered out during the triangulation using many robust probabilistic methods [4], [7]. Random sampling consensus or RANSAC is one of the most widely used methods. It works by randomly picking a small subset of the seed matches, using that subset to estimate the transformation function, and then using a larger subset to check the transformation function correctness [8].

In SfM, detected features in a sequence of images are required to be accurately matched despite the image transformation (rotation and/or scaling) and illumination changes. The matched features across the sequence of images allow for predicting the camera motion by estimating the calibration matrix as well as the fundamental matrix. Accordingly, the accuracy of these estimations and consequently the 3D reconstruction is directly affected by the robustness of the feature extraction algorithms and the accuracy of the feature matching algorithms [3]. Once feature correspondences between multiple views (images) are established, corresponding features are grouped into feature tracks. Corresponding features in each track are triangulated into a single 3D point. A sparse 3D point cloud is the result of triangulating corresponding features in all tracks [9].

More images can be incorporated into the current model using image registration. During image registration, new images are registered to the current model. This can be done by solving the Perspective-n-Point (PnP) problem. Given a set of correspondences between 3D points of an object (the current model) and their respective projections on the camera plane (of the new image), PnP problem can be solved to estimate the relative pose between the object and the camera by minimizing the reprojection error from 3D-2D point correspondences. To improve this incremental SfM pipeline, the reconstructed 3D point cloud is refined using Bundle Adjustment optimization technique resulting in a jointly optimal 3D point cloud [10].

The remaining part of this paper is organized as follows: Section II briefly covers the related work. Section III is a brief review of different feature extraction algorithms implemented for the experiment. Section IV describes the evaluation methodology proposed. Section V discusses the experimental results obtained. Section VI conclusion and future work reported.

II. Related Work

Up to the best of the author's knowledge and as shown in Table. I, published research evaluates feature extraction algorithms (particularly in SfM) in terms of feature extraction time, number of extracted features, descriptor size (storage), number of matched features, point cloud size (density), and reprojection error.

TABLE I. RELATED WORK

Author	**Algorithms**	**Factor**	**Winner**
Govender [3]	HCD, KLY, SIFT, SURF	Error	SIFT
Urban et al. [11]	AKAZE + M-SURF, ORB, SIFT, SURF, SURF + BinBoost	Time	SURF
Chien et al. [2]	AKAZE, ORB, SIFT, SURF	#Features	SURF
		Storage	AKAZE
		Time	ORB
Pusztai et al. [5]	AGAST, AKAZE, BRISK, FAST, GFTT, KAZE, MSER, ORB, SIFT, STAR, SURF	#Features (Inliers)	SURF
Schönberger et al. [12]	ConvOpt, DeepDesc, DSP-SIFT, LIFT, SIFT, SIFT-PCA, TFeat	Density	Varies
		Error	SIFT
		Storage	SIFT
		Time	SIFT
Cao et al. [10]	BRISK, KAZE, ORB, SIFT, SURF	Error (ROS)	SURF
		Time	BRISK
Gao et al. [9]	AKAZE, DeepCompare, LF-Net, ORB, SIFT, SuperPoint, SURF	Density	SURF
		Error	AKAZE
		Time	ORB
Yusefi et al. [7]	FAST, ORB, SIFT, STAR, SURF	Error	FAST
		Time	STAR

As shown in Table II, each algorithm has one or more parameters that affect its sensitivity in most cases. Using the default values for these parameters results in a quite different average number of extracted features, as shown in Table IV. Therefore, it may be unfair to use the number of extracted features as an evaluation factor of feature extraction algorithms without trying to tune the parameters of each algorithm. In addition, the extraction time depends on the number of extracted features and the descriptor type and size. Therefore, it may be unfair to evaluate feature extraction algorithms in terms of the extraction time without taking the number of extracted features (i.e., by averaging), descriptor type, and descriptor size into consideration. Moreover, there is a trade-off between the point-cloud size and the reprojection error. Therefore, it may be unfair to evaluate feature extraction algorithms in SfM based on the value of point-cloud size only or reprojection error only.

The scope of this paper is fairly evaluating the robustness of the feature extraction algorithms in SfM in terms of point cloud size and reprojection error regardless of the space and time efficiency.

III. Feature Detectors and Descriptors

Table II lists the fourteen feature extraction algorithms that were evaluated in this paper. A feature extraction algorithm can be designed for feature detection only, feature description only, or both feature detection and description. In the last case, feature detectors are combined with their own feature descriptors. In the first two cases, different types of feature descriptors can be paired with different kinds of feature detectors.

TABLE II. PARAMETERS OF FEATURE EXTRACTION ALGORITHMS

AKAZE	[13]	descriptor_size, descriptor_channels, threshold, nOctaves, nOctaveLayers
BEBLID*	[14]	n_bits, scale_factor
BRIEF*	[15]	bytes
BRISK	[16]	thresh, octaves
DAISY*	[17]	radius, q_radius, q_theta, q_hist, norm
FREAK*	[18]	patternScale, nOctaves
KAZE	[19]	threshold, nOctaves, nOctaveLayers
LATCH*	[20]	bytes, half_ssd_size, sigma
LUCID*	[21]	lucid_kernel, blur_kernel
ORB	[22]	nfeatures, scaleFactor, nlevels, edgeThreshold, firstLevel, WTA_K, patchSize, fastThreshold
SIFT	[23]	nfeatures, nOctaveLayers, contrastThreshold, edgeThreshold, sigma
SURF	[24]	extended, hessianThreshold, nOctaves, nOctaveLayers
TEBLID*	[25]	n_bits, scale_factor
VGG*	[26]	desc, isigma, scale_factor

Algorithms identified by * in Table II do not have a default feature detector. For such algorithms, FAST (Features from Accelerated Segment Test) feature detection algorithm is combined with them in the experimental work in Section V. FAST algorithm was proposed by Rosten et al. in 2006 for fast feature detection [27], [28]. Therefore, it is considered a viable choice for complementing those algorithms.

Feature descriptors can be categorized into two main categories: namely parameterized descriptors and binary descriptors. Each element in parameterized descriptors is a floating-point number (or any non-binary discretization of a floating-point number). On the other hand, each element in binary descriptors is a binary number (0 or 1). Typically, binary descriptors are derived using pixel-level comparisons that require minimal computational resources and result in a compact representation. Consequently, they became an attractive option for numerous contemporary applications [10]. Out of the fourteen feature extraction algorithms that were evaluated in this paper, only DAISY, KAZE, SIFT, SURF, and VGG are parameterized while the others are binary.

IV. Proposed Evaluation Methodology

In this section, the proposed methodology for fairly evaluating the robustness of feature extraction algorithms in SfM is presented. The proposed methodology is expressed using Python-like pseudo-code in Fig. 1 and Fig. 2. It is also illustrated by a block diagram in Fig. 3.

```
Algorithm Constrained 3D Reconstruction (I, feat_alg)
Input : sequence of images I
    feature extraction algorithm feat_alg
Output: point cloud pc
F = {}
for i in I:
    f = feat_alg.extract(i)
    F[i] = sorted(f,
        key = lambda item:-item.response
    )[:MAX_FEAT]
M = {}
for i in I:
    for j in I:
        if I == j:
            continue
        m = feat_alg.match(F[i],F[j])
        M[i,j] = sorted(m,
            key = lambda item:item.distance
        )[:MAX_MATCH]
return colmap. mapper(I,M)
```

Fig. 1. Proposed Algorithm for Constrained 3D Reconstruction.

Constrained 3D Reconstruction algorithm shown in Fig. 1 takes a sequence of images as well as a feature extraction algorithm and returns a point cloud. Constrained 3D reconstruction is required since the quality of the 3D reconstruction depends on not only the robustness of the detected features but also the number of detected features. In order to ensure fairness and due to the difficulty of optimizing the parameters of each algorithm to produce the same number of features, only a fixed number of features detected by each algorithm is passed to the subsequent phase, as illustrated by Constrained Feature Extraction block in Fig. 3. Selection of the detected features can be performed by sorting them in non-increasing order of their respective responses and selecting the features with the highest response. In addition, only a fixed number of corresponding features matched using each algorithm is passed to the subsequent phase, as illustrated by Constrained Feature Matching block in Fig. 3. Selection of the corresponding features can be performed by storing them in non-decreasing order of matching distance and selecting the correspondences with the smallest matching distance.

Instead of the *ratio test* proposed by Lowe in [23] in the matching phase, brute-force matcher with cross-checking was used to return only consistent pairs of features. A pair of features (*i*, *j*) is considered consistent if *j* is the nearest feature to *i* and *i* is the nearest to *j* in the feature space. The consistent pairs of matching features are then passed to the next phase, where sparse point clouds are reconstructed.

Size-Error Curves Generation algorithm shown in Fig. 2 takes a list of point clouds generated using different feature extraction algorithms for a specific dataset and reconstructs size-error curves. Such curves visualize the average reprojection error of different 3D reconstructions as a function of the point cloud size. Moreover, the use of size-error curves allows employing methods like the elbow method [29] and finding a suitable operating point on ROC curve [30] for carefully selecting and/or pruning a reconstructed point cloud. Size-error curves are generated by first creating a list of sample sizes. The points in each point cloud are sorted in non-decreasing order of their respective reprojection error and then sampled using every sample size. The size-error curve can be visualized by plotting the average reprojection error of the points in each sample against the sample size.

```
Algorithm Size-Error Curves Generation (PC)

Input : list of point clouds PC,
    each point cloud is composed of a list of
    3D points (x, y, z, e)
Output: list of dictionaries D,
    each dictionary is a map from point cloud
    size to average reprojection error
K = []
for pc in PC:
    K.append(len(pc))
K = sorted(K)

D = []
for pc in PC:
    d = {}
    e = sorted(pc.e)
    for k in K:
        if k > len(pc):
            break
        d[k] = sum(e[:k])/k
    D.append(d)

return D
```

Fig. 2. Proposed Algorithm for Evaluation of 3D Reconstruction.

V. Experimental Results

A. Datasets

To evaluate the robustness of feature extraction algorithms, a well-known and publicly available collection of datasets was used as a benchmark. The collection contains six different datasets of outdoor scenes of different architectural objects. All images were captured at 3072 × 2028 resolution and have been corrected for radial distortion. Each dataset name, number of images, and sample image are shown in Table III. More information about the dataset can be found in [31], [32].

TABLE III. THE SIX DATASETS USED IN THIS RESEARCH

Dataset	castle-P19	castle-P30	entry-P10	fountain-P11	Herz-Jesus-P8	Herz-Jesus-P25
Sample						
Size	19	30	10	11	8	25

B. Setup

An experiment was performed on a virtual machine with a 2 x Core Intel(R) Xeon(R) CPU @ 2.20 GHz and 13.00 GB of RAM provided by Google Collaborate Pro to evaluate the robustness of different feature extraction algorithms. OpenCV 4.6.0 was built with OPENCV_ENABLE_NONFREE as well as OPENCV_EXTRA_MODULES_PATH, while COLMAP 3.8 was built with the default configuration.

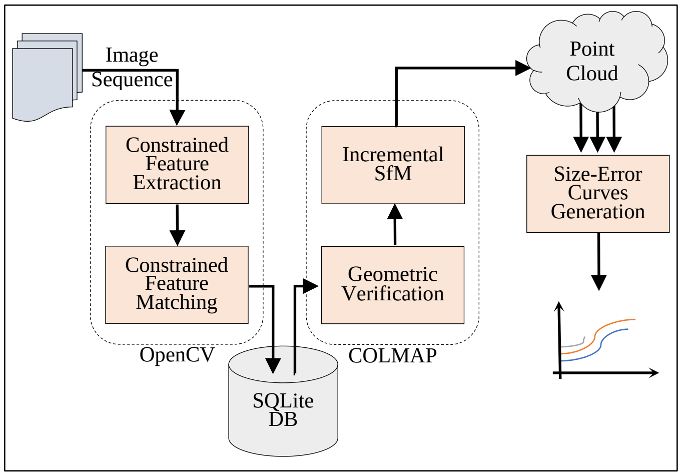


Image Sequence
Constrained Feature Extraction
Constrained Feature Matching
OpenCV
SQLite DB
Incremental SfM
Geometric Verification
COLMAP
Point Cloud
Size-Error Curves Generation

Fig. 3. Proposed Methodology for Fair Evaluation.

Feature detection and extraction subroutines contained within OpenCV library framework [33] were used with their default parameters provided by the library for Python. When a required parameter does not have a default value, the value recommended by the algorithm author(s) is used. Each detector is only combined with its own descriptor in this experiment, and detectors with no descriptor are combined with FAST due to its speed and availability in a real-time environment. OpenCV BFMatcher was used along with crossCheck flag and the default norm of each feature extraction algorithm for feature correspondence. COLMAP SfM pipeline [34], [35] was used on the CPU for spare 3D reconstruction. The complete pipeline is illustrated by Fig. 3.

Only the best 2000 detected features are passed to the next phase. In addition, only the best 500 corresponding matches are passed to the next phase. The two numbers were chosen empirically to suit almost all the feature extraction algorithms.

In order to connect OpenCV to COLMAP, an SQLite database is used for storing the following:

- Camera parameters.
- Images filenames.
- Coordinates of each extracted feature for each image.
- Indexes of each pair of corresponding features.

In order to reconstruct the corresponding sparse point clouds, a total of 14 x 6 SQLite databases were created and passed to COLAMP as command-line parameters.

C. Results and Discussion

During this experiment, 84 different point clouds were created. As shown in Section II, the projection error is the most common and obvious way to measure how good a point cloud is. V and the radar chart in Fig. 4 illustrate the reprojection error of each point cloud. By comparing the reprojection error resulting from different feature extraction algorithms on each dataset, the feature extraction algorithm with the smallest reprojection error can be considered the winner. Based on the result of this experiment, it is clear that FREAK resulted in the smallest reprojection error on most datasets (four out of six) and on average. Therefore, it can be considered the winner. However, the quality of the resulting point cloud is defined not only by the reprojection error but also by the point cloud size (density).

According to the results shown in VI and illustrated by the radar chart in Fig. 5, it is clear that the density of the point clouds generated using FREAK feature extraction algorithm is disappointing. It is almost the smallest for most of the datasets. This means that FREAK's low reprojection error comes at a cost (the sparsity of the point cloud).

Obviously, neither Fig. 4 nor Fig. 5 (alone or side-by-side) provide an accurate measure or visualization of the relative performance of different feature extraction algorithms in SfM. This raises the need for a measure or visualization that takes care of the quantity-quality trade-off.

On the other hand, using the size-error curves shown in Fig. 6, a true winner can be easily identified by observing the area under the curve (AUC). The feature extraction algorithm corresponding to the smallest AUC and adequate density can be considered a winner regardless of point cloud size and the average reprojection error. For instance, it is clear from Fig. 6 that SIFT is the winner in Herz-Jesus-P8 dataset even though it did not generate the point cloud with the largest size or the smallest average reprojection error.

TABLE IV. THE AVERAGE NUMBER OF EXTRACTED FEATURES PER DATASET FOR EACH FEATURE EXTRACTION ALGORITHM

AKAZE	**BEBLID**	**BRIEF**	**BRISK**	**DAISY**	**FREAK**	**KAZE**	**LATCH**	**LUCID**	**ORB**	**SIFT**	**SURF**	**TEBLID**	**VGG**
099888	163893	158462	081135	163893	159357	098562	158662	163893	257609	095820	359358	163893	163893

TABLE V. REPROJECTION ERROR USING DIFFERENT FEATURE EXTRACTION ALGORITHMS

	castle-P19	**castle-P30**	**entry-P10**	**fountain-P11**	**Herz-Jesus-P25**	**Herz-Jesus-P8**	**Average**
AKAZE	0.6175	0.6171	0.7233	0.6473	0.8661	0.6831	0.6924
BEBLID	**0.4798**	0.5159	0.5705	0.5404	0.6353	0.6204	0.5604
BRIEF	0.5139	0.5075	0.6058	0.5947	0.6840	0.6364	0.5904
BRISK	0.5619	0.7170	0.7034	0.7701	0.9119	0.8535	0.7530
DAISY	0.5291	0.5687	0.6357	0.5628	0.6956	0.5994	0.5986
FREAK	0.5850	**0.4956**	**0.4923**	0.4446	**0.4891**	**0.5331**	**0.5066**
KAZE	0.6271	0.6521	0.8231	0.6488	0.8392	0.7331	0.7206
LATCH	0.5643	0.5717	0.6984	0.7157	0.8138	0.7649	0.6881
LUCID							
ORB	0.7029	0.7575	0.7366	0.8767	0.8932	0.8264	0.7989
SIFT	0.5730	0.5777	0.5336	**0.3497**	0.6307	0.5575	0.5370
SURF	0.6112	0.5870	0.6773	0.4715	0.7324	0.6789	0.6264
TEBLID	0.5133	0.5203	0.5927	0.5431	0.6507	0.6049	0.5709
VGG	0.4966	0.5372	0.6016	0.5283	0.6463	0.5915	0.5669
Average	0.5674	0.5866	0.6457	0.5918	0.7299	0.6679	0.6315

TABLE VI. POINT CLOUD SIZE USING DIFFERENT FEATURE EXTRACTION ALGORITHMS

	castle-P19	**castle-P30**	**entry-P10**	**fountain-P11**	**Herz-Jesus-P25**	**Herz-Jesus-P8**	**Maximum**
AKAZE	2160	5869	1497	1375	3326	1212	5869
BEBLID	1559	4664	1393	1525	3036	1090	4664
BRIEF	1795	4694	1429	1473	3001	1087	4694
BRISK	1170	1229	1193	1275	2947	1088	2947
DAISY	2065	5656	**1664**	1642	3074	1145	5656
FREAK	1054	2460	1026	1228	2524	810	2524
KAZE	2151	**6177**	1591	1494	3609	**1229**	6177
LATCH	1825	5327	1412	1460	2984	1085	5327
LUCID							
ORB	1238	2246	1015	0877	2304	1046	2304
SIFT	**2240**	6118	1428	**1702**	**3614**	1197	6118
SURF	1338	4744	0934	1323	3350	1050	4744
TEBLID	1797	4528	1475	1549	3167	1140	4528
VGG	1740	4876	1468	1595	3068	1141	4876
Maximum	2240	6177	1664	1702	3614	1229	6177

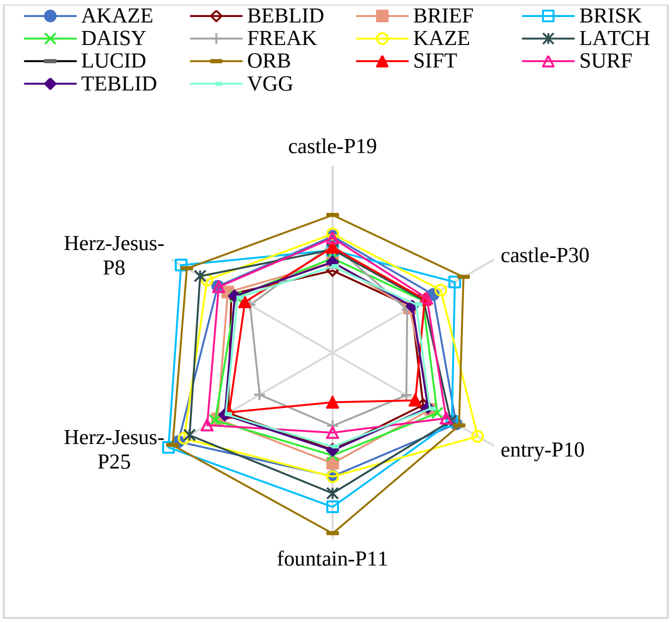


Fig. 4. Reprojection Error using different Feature Extraction Algorithms.

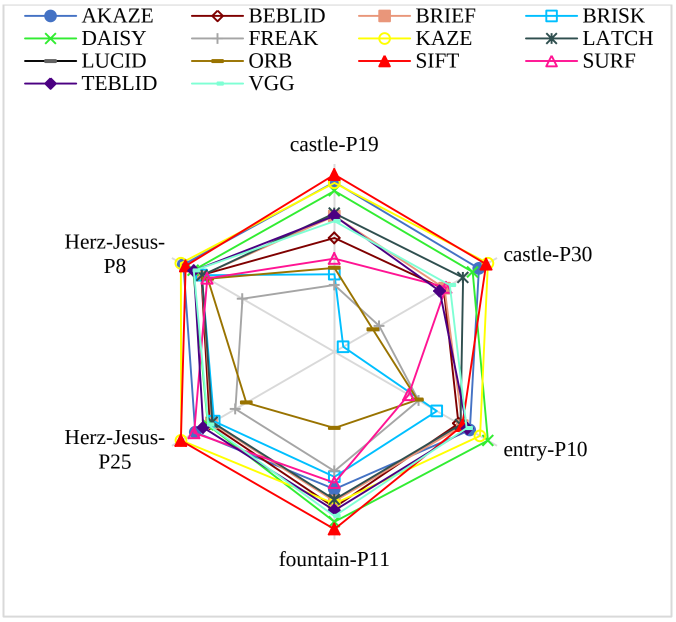
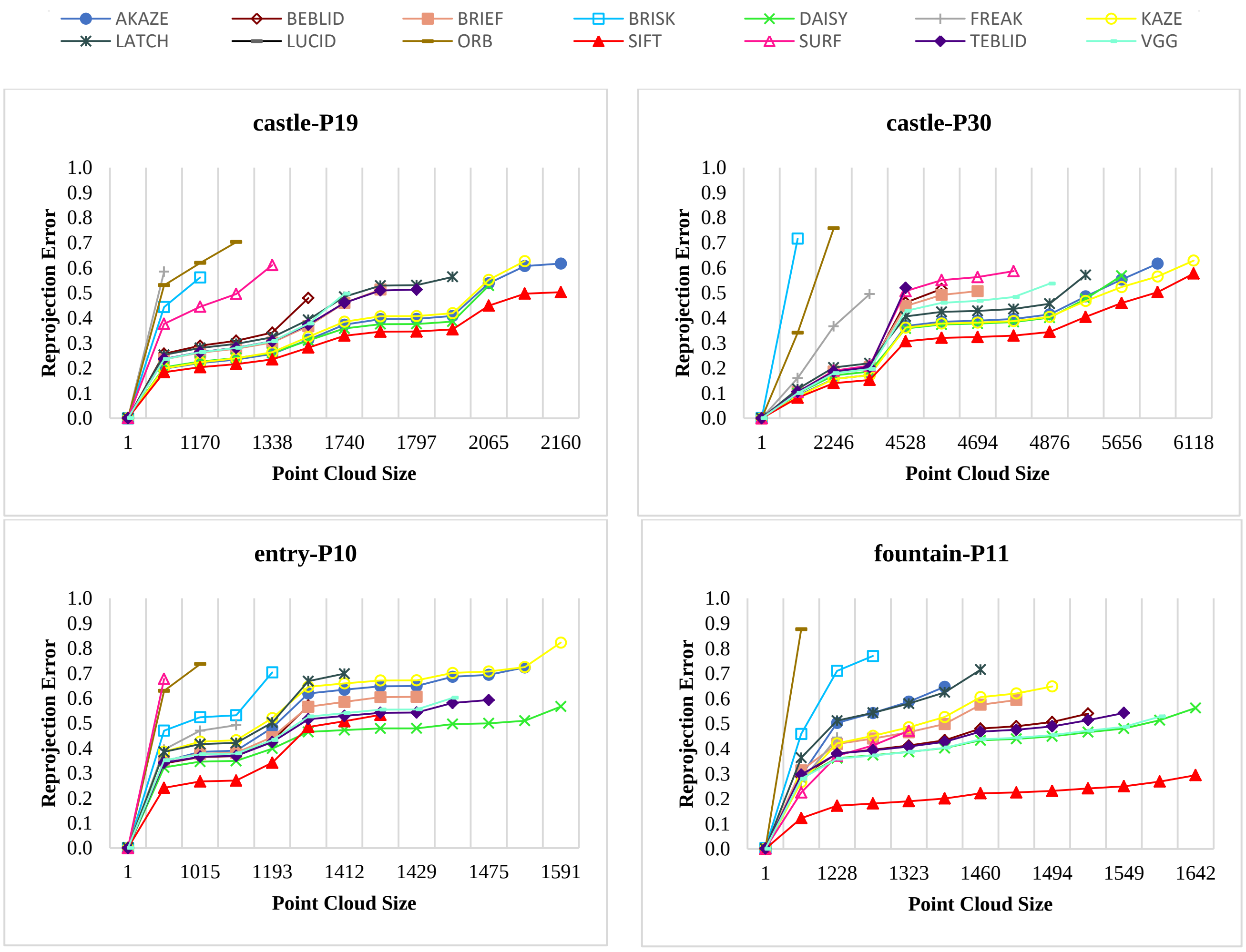


Fig. 5. Point cloud Size using different Feature Extraction Algorithms.



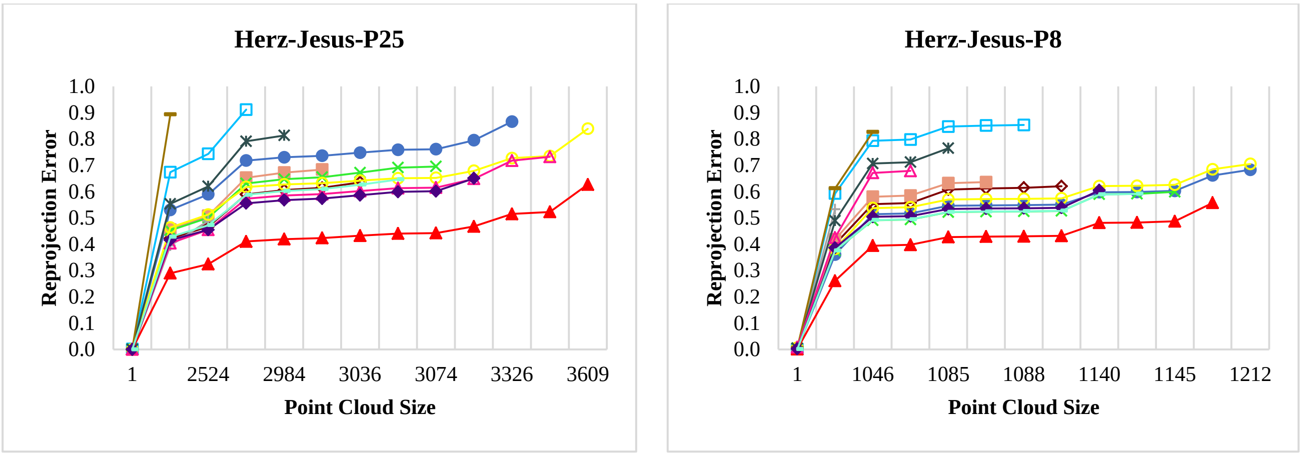


Fig. 6. Reprojection Error as a Function of the Point cloud Size for the Six Datasets.

VI. Conclusion and Future Work

In this research, fourteen different OpenCV feature extraction algorithms were evaluated in COLMAP SfM pipeline on six public datasets. Based on the experimental results reported in Section V, it can be concluded that the best feature extraction algorithm does not necessarily generate a point cloud with the minimum reprojection error nor the maximum number of points in SfM. So, the size-error curves proposed in this paper should be used for comparing different 3D reconstructions of a given dataset, as they are more transparent than using the value of point cloud size, reprojection error, or both.

The size-error curves demonstrate that SIFT feature extraction algorithm outperforms other algorithms in almost all six datasets. Our future work is to consider combining feature detectors other than FAST with descriptors that do not have a default detector and those that do. By applying the fair evaluation method proposed in this paper to the different combinations, a better combination of feature detector and feature descriptor for SfM may be found.

References

[1] S. A. K. Tareen and Z. Saleem, “A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK,” in 2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, iCoMET 2018 - Proceedings, 2018, vol. 2018-January. doi: 10.1109/ICOMET.2018.8346440.

[2] H. J. Chien, C. C. Chuang, C. Y. Chen, and R. Klette, “When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry,” in International Conference Image and Vision Computing New Zealand, 2016, vol. 0. doi: 10.1109/IVCNZ.2016.7804434.

[3] N. Govender, “Evaluation of Feature Detection Algorithms for Structure from Motion,” Csir, 2009.

[4] S. Bianco, G. Ciocca, and D. Marelli, “Evaluating the performance of structure from motion pipelines,” J Imaging, vol. 4, no. 8, 2018, doi: 10.3390/jimaging4080098.

[5] Z. Pusztai and L. Hajder, “Quantitative Comparison of Feature Matchers Implemented in OpenCV3,” in 21 st. Computer Vision Winter Workshop, Rimske Toplice, 2016.

[6] R. Szeliski, “Computer Vision : Algorithms and Applications 2nd Edition,” Springer, 2021.

[7] A. YUSEFI, A. DURDU, and C. SUNGUR, “Performance and Trade-off Evaluation of SIFT, SURF, FAST, STAR and ORB feature detection algorithms in Visual Odometry,” European Journal of Science and Technology, 2020, doi: 10.31590/ejosat.819735.

[8] M. A. Fischler and R. C. Bolles, “Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” Commun ACM, vol. 24, no. 6, 1981, doi: 10.1145/358669.358692.

[9] K. Gao et al., “Local feature performance evaluation for structure-from-motion and multi-view stereo using simulated city-scale aerial imagery,” IEEE Sens J, vol. 21, no. 10, 2021, doi: 10.1109/JSEN.2020.3042810.

[10] M. Cao et al., “Evaluation of local features for structure from motion,” Multimed Tools Appl, vol. 77, no. 9, 2018, doi: 10.1007/s11042-018-5864-1.

[11] S. Urban and M. Weinmann, “Finding a good feature detector-descriptor combination for the 2d keypoint-based registration of tls point clouds,” in ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2015, vol. 2, no. 3W5. doi: 10.5194/isprsannals-II-3-W5-121-2015.

[12] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, “Comparative evaluation of hand-crafted and learned local features,” in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017, vol. 2017-January. doi: 10.1109/CVPR.2017.736.

[13] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, “Fast explicit diffusion for accelerated features in nonlinear scale spaces,” in BMVC 2013 - Electronic Proceedings of the British Machine Vision Conference 2013, 2013. doi: 10.5244/C.27.13.

[14] I. Suárez, G. Sfeir, J. M. Buenaposada, and L. Baumela, “BEBLID: Boosted efficient binary local image descriptor,” Pattern Recognit Lett, vol. 133, 2020, doi: 10.1016/j.patrec.2020.04.005.

[15] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary robust independent elementary features,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2010, vol. 6314 LNCS, no. PART 4. doi: 10.1007/978-3-642-15561-1_56.

[16] S. Leutenegger, M. Chli, and R. Y. Siegwart, “BRISK: Binary Robust invariant scalable keypoints,” in Proceedings of the IEEE International Conference on Computer Vision, 2011. doi: 10.1109/ICCV.2011.6126542.

[17] E. Tola, V. Lepetit, and P. Fua, “DAISY: An efficient dense descriptor applied to wide-baseline stereo,” IEEE Trans Pattern Anal Mach Intell, vol. 32, no. 5, 2010, doi: 10.1109/TPAMI.2009.77.

[18] A. Alahi, R. Ortiz, and P. Vandergheynst, “FREAK: Fast retina keypoint,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2012. doi: 10.1109/CVPR.2012.6247715.

[19] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, “KAZE features,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012, vol. 7577 LNCS, no. PART 6. doi: 10.1007/978-3-642-33783-3_16.

[20] G. Levi and T. Hassner, “LATCH: Learned arrangements of three patch codes,” in 2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016, 2016. doi: 10.1109/WACV.2016.7477723.

[21] A. Ziegler, E. Christiansen, D. Kriegman, and S. Belongie, “Locally uniform comparison image descriptor,” in Advances in Neural Information Processing Systems, 2012, vol. 1.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in Proceedings of the IEEE International Conference on Computer Vision, 2011. doi: 10.1109/ICCV.2011.6126544.

[23] D. G. Low, “Distinctive image features from scale-invariant keypoints,” Int J Comput Vis, 2004.

[24] H. Bay, T. Tuytelaars, and L. van Gool, “SURF: Speeded up robust features,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2006, vol. 3951 LNCS. doi: 10.1007/11744023_32.

[25] I. Suarez, J. M. Buenaposada, and L. Baumela, “Revisiting Binary Local Image Description for Resource Limited Devices,” IEEE Robot Autom Lett, vol. 6, no. 4, 2021, doi: 10.1109/LRA.2021.3107024.

[26] K. Simonyan, A. Vedaldi, and A. Zisserman, “Learning local feature descriptors using convex optimisation,” IEEE Trans Pattern Anal Mach Intell, vol. 36, no. 8, 2014, doi: 10.1109/TPAMI.2014.2301163.

[27] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2006, vol. 3951 LNCS. doi: 10.1007/11744023_34.

[28] S. Y. El-Mashad and A. Shoukry, “Evaluating the robustness of feature correspondence using different feature extractors,” in 2014 19th International Conference on Methods and Models in Automation and Robotics, MMAR 2014, 2014. doi: 10.1109/MMAR.2014.6957371.

[29] R. L. Thorndike, “Who belongs in the family?,” Psychometrika, vol. 18, no. 4, 1953, doi: 10.1007/BF02289263.

[30] C. E. Metz, “Basic principles of ROC analysis,” Semin Nucl Med, vol. 8, no. 4, 1978, doi: 10.1016/S0001-2998(78)80014-2.

[31] C. Strecha, W. von Hansen, L. van Gool, P. Fua, and U. Thoennessen, “On benchmarking camera calibration and multi-view stereo for high resolution imagery,” in 26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2008. doi: 10.1109/CVPR.2008.4587706.

[32] A. Ley, R. Hänsch, and O. Hellwich, “Syb3r: A realistic synthetic benchmark for 3D reconstruction from images,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 9911 LNCS. doi: 10.1007/978-3-319-46478-7_15.

[33] G. Bradski, “The OpenCV Library,” Dr. Dobb’s Journal of Software Tools, 2000.

[34] J. L. Schönberger, E. Zheng, J. M. Frahm, and M. Pollefeys, “Pixelwise view selection for unstructured multi-view stereo,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 9907 LNCS. doi: 10.1007/978-3-319-46487-9_31.

[35] J. L. Schonberger and J. M. Frahm, “Structure-from-Motion Revisited,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, vol. 2016-December. doi: 10.1109/CVPR.2016.445.