Cumhuriyet Science Journal
CSJ

e-ISSN: 2587-246X
ISSN: 2587-2680

Cumhuriyet Sci. J., Vol.40-4 (2019) 958-966

Comparative Analysis of the Feature Extraction Performance of Augmented Reality Algorithms

Umut TOSUN[1*]

[1] Alanya Alaaddin Keykubat University, Faculty of Engineering, Department of Computer Engineering, Alanya, TURKEY

Received: 25.10.2019; Accepted: 21.11.2019 *http://dx.doi.org/10.17776/csj.638297*

Abstract. The algorithms that extract keypoints and descriptors in augmented reality applications are getting more and more important in terms of performance. Criterions like time and correct matching of points gain more impact according to the type of application. In this paper, the performance of the algorithms used to identify an image using keypoint and descriptor extraction is studied. In the context of this research, main criterion like the number of keypoints and descriptors that the algorithms extract, algorithm execution time, and the quality of keypoints and descriptors extracted are considered as the performance metrics. Same data stacks were used for obtaining comparison results. In addition to comparisons for a group of well-known augmented reality applications, the best performing algorithms for varying applications were also suggested. C++ language and OpenCV library were used for the implementation of the augmented reality algorithms compared.

Keywords: Augmented Reality, Image Processing, Key Point, Descriptor.

Artırılmış Gerçeklik Algoritmalarının Öznitelik Çıkarma Performanslarının Karşılaştırmalı Analizi

Özet. Artırılmış gerçeklik uygulamalarında kullanılan anahtar nokta ve öznitelik çıkaran algoritmalar performansları açısından önem teşkil etmektedirler. Uygulamanın türüne göre zaman, noktaların doğru eşleşmesi gibi kriterler önem kazanmaktadır. Bu makalede artırılmış gerçeklik uygulamalarında kullanılan ve bir resmi tanımak amacı ile resim üzerinde anahtar nokta ve öznitelik bulunması için uygulanan algoritmaların performansları incelenmiştir. Çalışma kapsamında, algoritmaların çıkarabildiği anahtar nokta sayısı, öznitelik sayısı, algoritmanın çalışması sırasında geçen süre, iki resmin eşleştirilmesi sırasında çıkartılan anahtar nokta ve özniteliklerin kaliteleri gibi ana kriterler incelenmiştir. Karşılaştırma sonuçlarının elde edilmesinde, aynı veri kümeleri kullanılmıştır. Bu çalışmada, iyi bilinen bir grup artırılmış gerçeklik algoritması incelenerek performanslarının karşılaştırılmasının yanında, farklı uygulamalar için kullanılabilecek algoritmalar hakkında da önerilerde bulunulmuştur. Artırılmış gerçeklik algoritmalarının karşılaştırılması için C++ dili ve OpenCV kütüphaneleri kullanılmıştır.

Anahtar Kelimeler Artırılmış Gerçeklik, Görüntü İşleme, Anahtar Nokta, Öznitelik.

1. INTRODUCTION

Augmented reality is the result of combining real data and computer generated sound, images, graphics and location information in the world we live in. [1]. In other words, it is the enrichment of reality with virtual data in computer environment [2, 3]. Various methods

* Corresponding author. *Email address: umut.tosun@alanya.edu.tr*

are used to recognize the actual images in the virtual environment. In order to recognize an image from the real world, there are qualified points on the image.

Descriptors of a feature are extracted with respect to some of the basic properties of this point (pixels) and other points around it. The dots on the image that have the feature value for that image are then used for virtual recognition of this image. Thus, in the virtual environment, a person has the values of the picture that they want to be recognized. The user can compare the picture that he has previously extracted and wanted to recognize with the other pictures taken from the camera. A “threshold" value should be determined during the matching phase. Otherwise, if the features taken from the camera and extracted from a picture that we do not want to recognize actually match the features of the picture that is intended to be recognized, a wrong match will be made. After setting the threshold value, it is possible to match the image to be recognized with the images taken from the camera. One of the methods used to extract key points and features on the image is the Oriented Fast and Rotated Brief (ORB) algorithm [4].

ORB is a binary algorithm. Within the scope of this paper, one of the reasons why ORB algorithm is compared with other methods is the opinion that it is faster than basic algorithms in the literature which extract some key points and features. The fact that ORB is based on binary descriptor and that pairing between two pictures happens while performing matching, supports the idea that it is faster than other methods.

In the literature, fps (frame per second), the number of key points extracted on one image, the number of features extracted on one image and the number of features matched correctly between the two images are used to compare the algorithms used in the extraction of key points and features [5-10]. This work will be based on the criteria mentioned. At the end of this study, the algorithms used for augmented reality an developed for the recognition of an image will be compared and their performance will be benchmarked. At the end of the study, the algorithms used for augmented reality and developed for the recognition of an image will be compared and their performance will be benchmarked. Moreover, some suggestions on the usage areas will be made.

2. COMPARED ALGORITHMS

Keypoint is a pixel that has a specific meaning on an image. In calculating the key point, various algorithms can be applied depending on the type of application. The ORB algorithm [4] uses the FAST [11, 12] algorithm in the background when calculating the key points on the image. The FAST algorithm basically calculates the key points by targeting the corners on the image. There may be many meaningless pixels on the picture. These insignificant pixels cause loss of performance within augmented reality. Therefore, it is important to identify and process key points. However, in some cases, key points may not make sense alone. Feature values should be calculated while matching the key points on the picture taken from the real world with the picture that is required to be recognized within the context of augmented reality. Thus, when pairing the two images, more accurate matches can be made with the feature values of the key points extracted from the objects to be recognized.

Feature can be defined as scalable and observable information obtained from the image [13, 14]. Feature extraction removes a significant set of features by discarding unnecessary information [15]. Feature extraction aims to reduce processing time by reducing the size of high-dimensional data. Using this data as it is in image processing applications increases processing complexity. Feature extraction is an important part of augmented reality applications in terms of application performance [16]. It aims to increase the recognition success by expressing the information of the pattern in the smallest dimension with the most prominent features [17]. Dimension reduction is performed by extracting unnecessary information that is irrelevant to the pattern and obtaining specific properties. This process aims to create a more selective set by obtaining a subset of the feature set using different methods. The algorithms to be compared in this study will perform feature extraction. As a result of feature extraction, algorithms will be compared according to criteria such as time, correct matching, number of features.

2.1. Method Oriented Fast & Rotated BRIEF

Oriented Fast and Rotated Brief (ORB) [4], proposed by Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski, is an effective algorithm alternative for SIFT [18] or SURF [19]. ORB is basically a combination of the FAST [11, 12] key point and BRIEF [20] descriptor, but it also incorporates many performance-enhancing modifications. Oriented FAST and rotated BRIEF techniques are interesting because of their good performance and low costs. ORB first finds key points using FAST, then applies the Harris corner measure to find the top N points between them [21]. ORB also uses the pyramid to produce multi-scale features. With the method called rBRIEF [22], it searches for all possible binary tests to find high variance, as well as averages close to 0.5, as well as non-correlated ones.

2.2. Scale-Invariant Feature Transform

Scale-Invariant Feature Transform (SIFT) [18] is an algorithm proposed by David Lowe for identifying and describing regional features in an image. The key points are extracted by the SIFT sensor and their descriptors are calculated by the SIFT descriptor. The SIFT sensor or SIFT descriptor can also be used independently of each other (such as calculating key points without descriptors or calculating descriptors without special key points) [23]. Along with linked descriptors, SIFT has created a new field of research on image-based matching and recognition with many application areas. Multi-image matching, object recognition, object category classification and robotics are among the known uses of this algorithm [24].

2.3. Speeded Up Robust Feature

The Speed Up Robust Feature (SURF) is a powerful regional feature sensor presented by Herbert Bay and friends, which can be used in computer image tasks such as object recognition or 3D reconstruction [19]. SURF is partly inspired by the SIFT [18] descriptor, but the standard versions of SURF work much faster than SIFT. It is also stated that SURF is more powerful than SIFT against different image transformations. SURF is based on the sum of 2D Haar small wave elements and enables the effective use of integral images [19]. In addition, SURF was advanced over SIFT by applying box filter approximation to the convolution kernel of the Gaussian derivative operator. Experiments on camera calibration and object identification also reveal that SURF has a large potential for computer vision applications [25].

2.4. Fast Retina Keypoint

Fast Retina Keypoint (FREAK) is a key point descriptor presented by Alexandre Alahi and friends. [26]. The creation of FREAK was inspired by the human visual system and the retina. The cascading of binary sequences is calculated by effective comparison of image densities on the retinal sampling pattern. In their experiments, Alexander Alahi and colleagues showed that FREAK was generally more powerful and faster in computing with lower memory load than SIFT [18], SURF [19] or BRISK [27]. FREAK is therefore considered to be a competitive alternative to existing algorithms, especially for embedded applications [27].

3. EXPERIMENTS AND RESULTS

In order to analyze the performance of the algorithms, a framework has been developed in Visual Studio 2012 using C ++ language and OpenCV library. This application can run the algorithms ORB [4], SURF [19], SIFT [18] and FREAK [26]. The application first reads a fixed image and extracts key points and features from that image. The key values and features of this image will then be used for comparison for each image (frame) taken from a video taken with the camera. In the next step, the application reads a video for use in comparing algorithms. The prepared video is a video of the first fixed picture taken from different heights and angles.

The prepared video is used separately for each algorithm. Thus, it was considered to obtain the correct values. After reading the prepared video, the application takes individual pictures (frames). The key points and feature values are extracted from these images (frames) as in the fixed image. These extracted feature values are then assigned to a matching function with the feature values extracted from the fixed image. This allows you to see if the features match correctly. The application calculates the duration of each function for all algorithms.

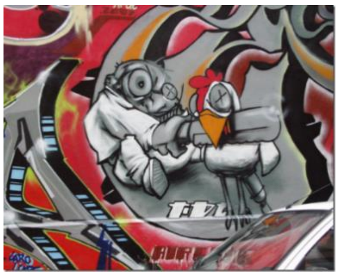
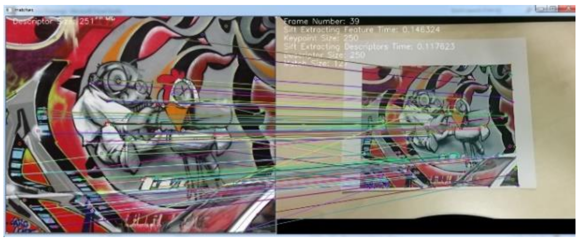


Figure 1. Fixed Image with Key Points Features Used in Testing

The image to be used as a fixed image is shown in Figure 1 and it is taken to video to be used in augmented reality subjects. In the testing phase and taken into the video is an image used in augmented reality subjects. The number of edges on the image, the plurality of curves, the richness of key points and features are the main reasons for the selection of said image.



```
Descriptor Size: 251
Frame Number: 39
Sift Extracting Feature Time: 0.146324
Keypoint Size: 250
Sift Extracting Descriptors Time: 0.117623
Descriptor Size: 250
Match Size: 14[illegible]
```

Figure 2. Screen Shot Taken During Application Run

During the operation of the application, key points and features are extracted from the fixed picture (Figure 1) and the pictures are taken from the video (frame) to be given to the matching function. In addition, key points that provide the threshold value are shown (Figure 2). Moreover, the number of the picture (frame) taken from the video, the number of key points and features that the algorithm generates for that picture (frame), and the number of features matching the threshold value are shown. There are 576 images (frames) in the video used to compare the algorithms. The threshold values 50, 100, 120 and 150 were used in the test procedures. The bit-based features are included in the comparison process, and as a result, the fs below the threshold value are calculated. Features below the threshold value correspond to a more accurate comparison result. As the threshold value increases, the number of matching features increases for algorithms.

As can be seen from the results in Table 1, the SIFT algorithm is the longest-running algorithm when the threshold is 50. However, the SIFT algorithm also provides the most accurate matchings. ORB algorithm is the fastest working algorithm according to the results. However, in the matching of features, SIFT and SURF algorithms gave worse results. From the results, the FREAK algorithm is both relatively slow and has poor results in the feature matching phase. One of the main reasons why the SIFT and SURF algorithms are slower than the ORB algorithm is the time it takes to extract key points and features. In Figure 3, the application algorithm is run for threshold 50 and the slowest running algorithm is observed as SIFT. ORB is the fastest completing algorithm. While the SIFT algorithm takes 1.182 seconds, the ORB algorithm performs key point and feature extraction in an average of 0.016 seconds, or the SURF algorithm it is 0.172 seconds and for the FREAK algorithm it is 0.655 seconds. Figure 4 shows the key and feature extraction times of the algorithms for threshold 50. According to the results, while the fastest running algorithm is ORB, the slowest running algorithm is SIFT. Figure 5 shows that the SIFT algorithm is most prominent with matching. The SURF algorithm has the most features matching after SIFT. Because the threshold value is 50; while the ORB algorithm performed 0.458 matching in an average picture (frame), the FREAK algorithm performed 0 matching. As can be seen from the results in Table 2, when the threshold value is given 120, the algorithm that runs the slowest and performs the highest number of matches, such as the results obtained at the threshold value 50, is SIFT.

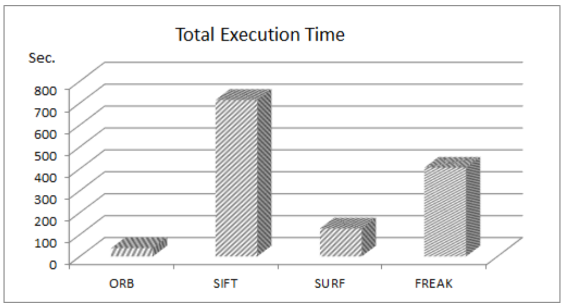


Figure 3: Total Execution Time for Threshold Value 50.

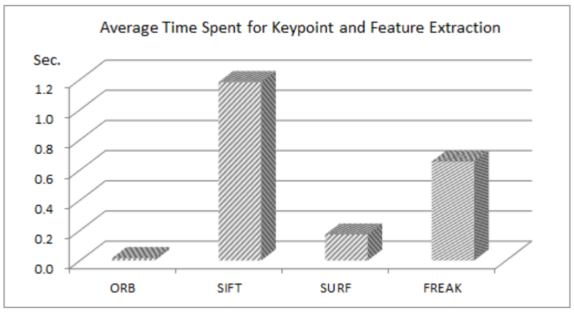


Figure 4: Average Time Spent for Key Point and Feature Extraction for Threshold Value 50

Table 1: Algorithm Performance for Threshold Value 50

	ORB	SIFT	SURF	FREAK
Base image keypoint size:	NULL	251	268	251
Base image time for keypoint extracting (seconds):	NULL	0.761	0.114	0.764
Base image descriptor size:	253	251	268	227
Base image time for descriptor extracting (seconds):	0.018	0.496	0.113	0.085
Base Image Total Feature & Descriptor Extracting Time (seconds):	0.018	1.257	0.227	0.849
Total Time (seconds):	38.120	714.094	127.543	404.836
Average Execute Time (seconds):	0.066	1.239	0.221	0.702
Total Feature Extracting Time (seconds):	NULL	393.397	52.961	371.75
Average Feature Extracting Time (seconds):	NULL	0.682	0.091	0.645
Total Descriptor Extracting Time (seconds):	9.259	287.975	46.550	5.821
Average Descriptor Extracting Time (seconds):	0.016	0.499	0.080	0.010
Total Feature & Descriptor Extracting Time (seconds):	9.259	681.372	99.511	377.571
Average Feature & Descriptor Extracting Time (seconds):	0.016	1.182	0.172	0.655
Total Match Time (seconds):	5.075	6.415	4.773	4.574
Average Match Time (seconds):	0.008	0.011	0.008	0.007
Total Draw Time (seconds):	7.216	9.625	6.895	5.895
Average Draw Time (seconds):	0.012	0.016	0.011	0.010
Average Match Size:	0.458	68.151	17.644	0

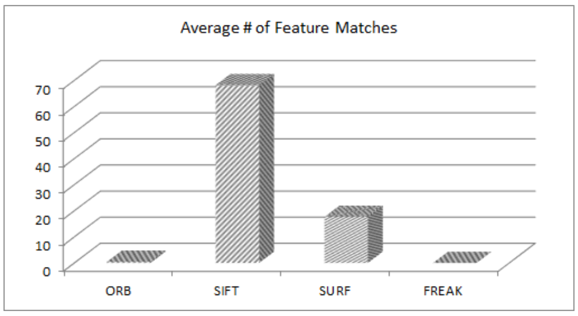


Figure 5: Average Number of Feature Matches for Threshold Value 50.

The ORB has more number of matches than SURF, according to the results observed at the threshold 50. ORB is seen to run faster than the other three algorithms. When matching numbers are compared, SURF has more feature matching numbers than other algorithms. FREAK was unable to match the feature in addition to running slowly. According to the results in the threshold value 50, the number of SURF algorithm matching features decreased. In Figure 6, the application algorithm threshold is run for 100 and the slowest running algorithm is observed as SIFT. ORB was the fastest completing algorithm. ORB completes the application in 38.541 seconds, while SIFT completes the application in 701.598 seconds (~ 11 minutes). For SURF, this time is 129.537 seconds, while FREAK finishes the application in 405.680 seconds. In order to obtain the results shown in Figure 7, the application worked with the threshold 100. SIFT takes an average of 1.159 seconds to extract key points and features from a picture (frame). This time directly affects the performance of the application. ORB is the fastest algorithm, as in the results with a threshold of 50, and takes an average of 0.015 seconds to extract key points and features from an image. SURF performs the processing in an acceptable time of 0.169 seconds. FREAK, like SIFT, has been working for a long time to affect performance. In Figure 8, the application threshold is run for 100 and the features extracted by the algorithms are compared with the features extracted from the base image and divided by the total number of frames, the average feature matching numbers are obtained. SIFT demonstrates the advantage of slow operation here. SIFT obtained an average number of feature matching of 123.11 images from the end of the video. ORB and SURF have very close matches. The SURF gave results close to the threshold value 50, while ORB increased the number of feature matching from 0.458 to 18.946.

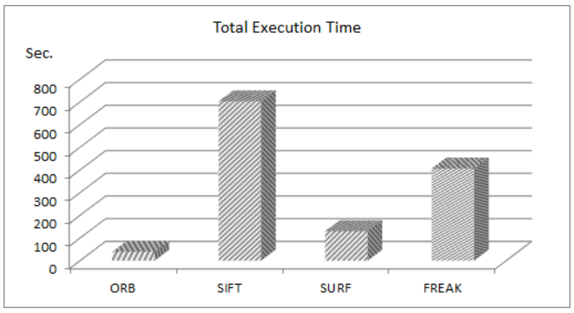


Figure 6: Total Execution Time for Threshold Value 100.

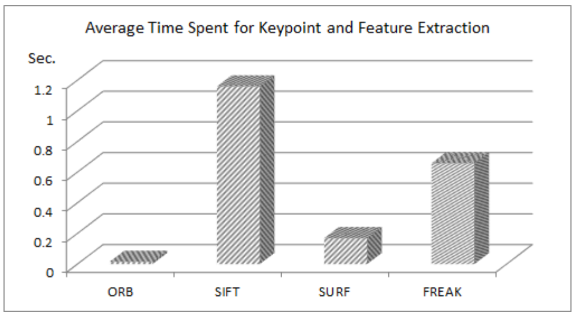


Figure 7: Average Time Spent for Key Point and Feature Extraction for Threshold Value 100

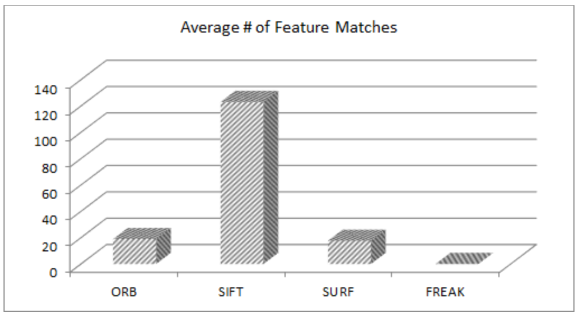


Figure 8: Average Number of Feature Matches for Threshold Value 100

As can be seen from the results in Table 2, when the threshold value is given 120, the slowest running algorithm is the SIFT as in the other threshold values. ORB, as the fastest running algorithm, has also increased the number of feature matchings obtained at threshold 100. FREAK was unable to match the feature in addition to running slowly. SURF completed the implementation within a reasonable time and again achieved an acceptable number of matches. In Figure 9, the application threshold is run for 120 and the slowest running algorithm is observed as SIFT. ORB is the fastest completing algorithm. ORB completed the application in 42,048 seconds, while SIFT completed the application in as long as 719.802 seconds (~ 11 minutes). SURF has completed the application in a suitable time of 125.222 seconds. FREAK completed the application in 423,282 seconds.

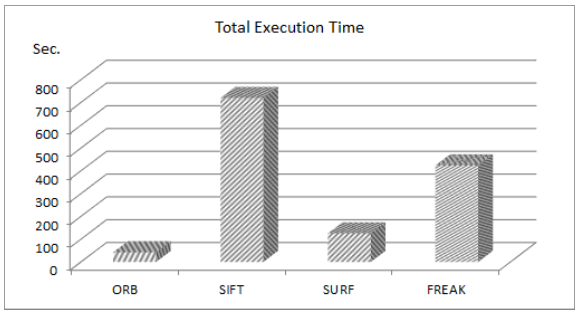


Figure 9: Total Execution Time for Threshold Value 120.

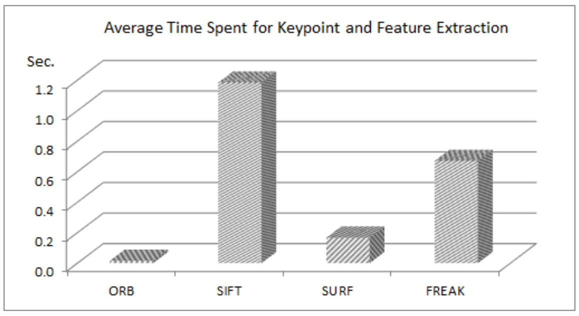


Figure 10: Average Time Spent for Key Point and Feature Extraction for Threshold Value 120

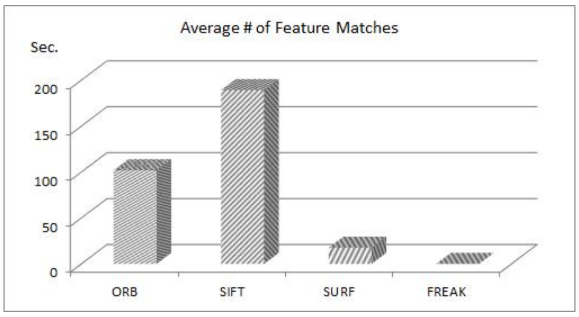


Figure 11: Average Number of Feature Matches for Threshold Value 120

Table 2: Algorithm Performance for Threshold Value 120.

	ORB	SIFT	SURF	FREAK
Base image keypoint size:	NULL	251	268	251
Base image time for keypoint extracting (seconds):	NULL	0.782	0.088	0.079
Base image descriptor size:	253	251	268	227
Base image time for descriptor extracting (seconds):	0.018	0.512	0.073	0.085
Base Image Total Feature & Descriptor Extracting Time (seconds):	0.018	1.294	0.161	0.864
Total Time (seconds):	42.048	719.802	125.222	423.282
Average Execute Time (seconds):	0.073	1.249	0.217	0.734
Total Feature Extracting Time (seconds):	NULL	390.273	50.569	380.600
Average Feature Extracting Time (seconds):	NULL	0.677	0.087	0.660
Total Descriptor Extracting Time (seconds):	9.429	291,465	45.786	5.875
Average Descriptor Extracting Time (seconds):	0.016	0.506	0.079	0.010
Total Feature & Descriptor Extracting Time (seconds):	9.429	681.738	96.355	386.475
Average Feature & Descriptor Extracting Time (seconds):	0.016	1.183	0.167	0.670
Total Match Time (seconds):	5.972	5.870	4.389	5.136
Average Match Time (seconds):	0.010	0.010	0.007	0.009
Total Draw Time (seconds):	11.876	15.504	7.010	6.026
Average Draw Time (seconds):	0.020	0.026	0.012	0.010
Average Match Size:	101.916	189.159	17.644	0.000

In Figure 10, the application is run and divided by the total number of frames (frames), the average time taken for a picture taken from the video is calculated for the key point and feature. While ORB completed key-point and feature extraction operations in a very small average time of 0.016, SIFT performed this operation on average for 1.183 seconds for an image, which means that an SIFT-powered application would run slowly. SURF completes key point and feature extraction in an acceptable average time of 0.167 seconds. FREAK completes key point and feature extraction in approximately 0.670 seconds. In Figure 11, the application is operated for the threshold 120. ORB increased the number of feature matches from 18.946 to 101.916 compared to the threshold 100. ORB removes an average of 253 features from an image and matches 101,916 of these features. SIFT can match an average of 189,159 features. SURF has maintained the 17,644 feature matching counts of threshold 100 for threshold 120. FREAK did not match any features. As the results in Table 3 show, when the threshold is set to 150, the slowest running algorithm is SIFT. ORB, as the fastest running algorithm, has increased the number of feature matches at threshold 120. FREAK has a very low number of features as well as slow operation. SURF completed the application within an acceptable time and matched the number of features obtained at the threshold 120.

Table 3: Algorithm Performance for Threshold Value 150.

	ORB	SIFT	SURF	FREAK
Base image keypoint size:	NULL	251	251	251
Base image time for keypoint extracting (seconds):	NULL	0.782	0.759	0.766
Base image descriptor size:	253	251	251	227
Base image time for descriptor extracting (seconds):	0.018	0.512	0.493	0.086
Base Image Total Feature & Descriptor Extracting Time (seconds):	0.018	1.294	1.252	0.852
Total Time (seconds):	49.391	719.802	699.518	412.129
Average Execute Time (seconds):	0.085	1.249	1.214	0.715
Total Feature Extracting Time (seconds):	NULL	390.273	371.997	378.546
Average Feature Extracting Time (seconds):	NULL	0.677	0.645	0.657
Total Descriptor Extracting Time (seconds):	9.241	291,465	286.577	5.875
Average Descriptor Extracting Time (seconds):	0.016	0.506	0.497	0.010
Total Feature & Descriptor Extracting Time (seconds):	9.241	681.738	658.574	384.421
Average Feature & Descriptor Extracting Time (seconds):	0.016	1.183	1.143	0.667
Total Match Time (seconds):	8,017	5.870	5.096	4.896
Average Match Time (seconds):	0.013	0.010	0.008	0.008
Total Draw Time (seconds):	18.369	15.504	18.826	6.104
Average Draw Time (seconds):	0.031	0.026	0.032	0.010
Average Match Size:	248.876	189.159	250.737	0.029

4. CONCLUSIONS

According to the results obtained, ORB is the fastest working algorithm. SIFT has the highest value in feature extraction criteria. Increasing the threshold value causes the fall of feature matching quality. In other words, increasing the threshold value causes the number of matching features generated by the algorithms to increase. As a result of this work, it was observed that SIFT and SURF performed more accurate feature matches. ORB has better results than the other three algorithms in terms of FPS (frame per second) time. In addition to being a slow algorithm like SIFT, FREAK is not successful in feature matching. Although SURF is relatively slow compared to ORB, it cannot provide feature matches such as SIFT. If it is an application where the simultaneous pictures taken from a camera are processed or pictures taken from a video are processed; extracting features for each image may cause a slowdown. If the desired number of features is matched on the picture, the feature may not be matched again on the next picture to be processed. Instead, features matched in the previous picture can be given to the follow-up phase depending on the type of application; thus, time can be saved as the follow-up phase runs faster than the feature matching phase. At this point, it is important to note that the key points that are processed as a result of feature matching during the follow-up phase are correctly matched key points. As a result, ORB works faster than the other three algorithms compared, and fewer features match with SIFT and SURF. Algorithm selection with respect to the application area of the augmented reality application to be implemented would be beneficial. For example, if a mobile application is to be realized, time will be an important criterion and ORB will give a good performance in this regard. SIFT or SURF will be more appropriate in high performance applications where time is not critical and correct feature matching is more important.

REFERENCES

[1] Graham M., Zook M., and Boulton A., Augmented Reality in Urban Places: contested content and the duplicity of code, Trans. Inst. Br. Geogr., 38-3 (2013) 464-479.

[2] Steuer J., Defining Virtual Reality: Dimensions Determining Telepresence, J. Commun., 42-4 (1992) 73-93.

[3] If You're Not Seeing Data You're Not Seeing, Wired.https://www.wired.com/2009/08/augmented-reality/. Retrieved October 25 (2009).

[4] Rublee E., Rabaud V., Konolige K., and Bradski G., ORB An Efficient Alternative to SIFT or SURF, Proceedings of 7th IEEE International Conference on Computer Vision, (2011) 2564-2571.

[5] Wagner D., Reitmayr G., Mulloni A., Drummond T., and Schmalstieg D., Pose Tracking from Natural Features on Mobile Phones, Proceedings of 7th IEEE and ACM International Symposium on Mixed and Augmented Reality, (2008) 125-134.

[6] Wagner D., Schmalstieg D., and Bischof H., Multiple Target Detection and Tracking with Guaranteed Framerates on Mobile Phones, International Symposium on Mixed and Augmented Reality, (2009) 57-64.

[7] Wagner D., Mulloni A., Langlotz T., and Schmalstieg D., Real-Time Panoramic Mapping and Tracking on Mobile Phones, Proceedings of IEEE Virtual Reality, (2010) 211-218.

[8] Klein G. and Murray D., Parallel Tracking and Mapping on a Camera Phone, Proceedings of 8th IEEE International Symposium on Mixed and Augmented Reality, (2009) 83-86.

[9] Ta D. N., Chen W. C., Gelfand N., and Pulli K., SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (2009) 2937-2944.

[10] Takacs G., Chandrasekhar V., Tsai S., Chen D., Grzeszczuk R., and Girod B., Rotation-invariant fast features for large-scale recognition and real-time tracking, Signal Process. Image, 28-4 (2013) 334-34.

[11] Rosten E. and Drummond T., Machine learning for high-speed corner detection, European Conference on Computer Vision, (2006) 430-443 .

[12] Rosten E., Porter R., and Drummond T., Faster and better: A machine learning approach to corner detection, IEEE T. Pattern Anal., 32-1 (2010) 105–119.

[13] Trzcinski T., Christoudias M., Lepetit V., and Fua P., Learning Image Descriptors with the

Boosting-Trick, Advances in Neural Information Processing Systems, (2012) 1-9.
[14] Winder S. and Brown M., Learning Local Image Descriptors, IEEE Conference on Computer Vision and Pattern Recognition, (2007) 1-8.
[15] Brown M., Hua G., and Winder S., Discriminative Learning of Local Image Descriptors, IEEE T. Pattern Anal., 33-1 (2011) 43-57.
[16] Ke Y. and Sukthankar R., PCA-SIFT: A More Distinctive Representation for Local Image Descriptors, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (2004) 506-514.
[17] Simonyan K., Vedali A., and Zisserman A., Descriptor Learning Using Convex Optimisation, European Conference on Computer Vision (2012), 243-256.
[18] Lowe D. G., Distinctive Image Features from Scale-Invariant Keypoints, Int. J. Comput. Vis., 60-2 (2004) 91-110.
[19] Bay H., Ess A., Tuytelaars T., and Gool L.V., Speeded-Up Robust Features (SURF), Comput. Vis. Image Und., 110-3 (2008), 346-359.
[20] Calonder M,, Lepetit V, Strecha C, and Fua P. Brief: Binary Robust Independent Elementary Features, European Conference on Computer Vision, Heraklion, (2010) 778-792.
[21] Harris C. and Stephens M., A combined corner and edge detector, Fourth Alvey Vision Conference, (1988) 147-151.
[22] Huang W., Wu L. D., Song H. C., and Wei Y. M. “RBRIEF: a robust descriptor based on random binary comparisons”. IET Comput. Vis,, 7-1 (2013) 29-35.
[23] Scale Invariant FeatureTransform, Scholarpedia.http://www.scholarpedia.org/article/SIFT. Retrieved October 18, 2013.
[24] Scale Invariant Feature Transform (SIFT), VLFeat. http://www.vlfeat.org/api/sift.html. Retrieved October 18, 2013.
[25] Schaeffer C., A Comparison of Keypoint Descriptors in the Context of Pedestrian Detection: FREAK vs. SURF vs. BRISK (2013).
[26] Alahi A., Ortiz R., and Vandergheynst P., FREAK: Fast Retina Keypoint, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (2012) 510-517.
[27] Leutenegger S., Chli M., and Siegwart R. Y., BRISK: Binary Robust invariant scalable keypoints, Proceedings of the IEEE International Conference on Computer Vision, (2011) 2548-2555.