J Inno Sci Eng,2022, 6(2):259-278
https://doi.org/10.38088/jise.1061166

Research Article

A Comparative Study on the Performance Analysis of Feature Extractors Used in Augmented Reality Applications under Various Image Conditions

Ceren Akman [1*], Ergun Gumus [2]

[1] *Kirsehir Ahi Evran University, Department of Computer Engineering, Kirsehir, Turkey*
[2] *Bursa Technical University, Department of Computer Engineering, Bursa, Turkey*

Abstract

Until today, various approaches have been proposed in order to create Augmented Reality (AR) environment where virtual-real integration takes place. One of these approaches is vision-based model, and it is divided into two branches as Marker-Based Augmented Reality (MBAR) and Markerless Augmented Reality (MAR). In the use of MBAR model, a reference image is introduced to the system before, and when this image enters the camera view, an AR environment is created. However, in MAR model, no image is introduced to the system before. Instead, it uses natural characteristics present in the image, such as edges, corners, and geometrical shapes to create an AR environment. In order to use MAR model, it is necessary to use algorithms which require high processing power and memory capacity. Within the scope of this study, MAR model was chosen as reference and an evaluation on combinations of descriptive extractors (such as ORB, SIFT, and SURF) and matchers (such as Bruteforce, Bruteforce-Hamming, and Flannbase) was presented. In this context, it was aimed that we could obtain knowledge about i) the number of key points and detection time with the use of different descriptor extractors, and ii) the number of matching key points and the amount of positional deviation of a virtual object placed on a real world scene with the use of different matchers. In line with this goal, analyses were made, using different image scales and brightness levels on both PC and mobile platforms. Results showed that, for both platforms, combinations using ORB method could work faster with less deviation than the combinations using other methods in all conditions. In addition, RANSAC algorithm was also used to reduce the total mean deviation ratio, and it was seen that the rate could be reduced from 70% to 4.5%..

Keywords: Augmented reality, Markerless augmented reality, Performance analysis, ORB, SIFT, SURF.

Cite this paper as: Ceren A. and Ergun G. (2022). A comparative study on the performance analysis of feature extractors used in augmented reality applications under various image conditions, 6(2):259-278.

*Corresponding author:
Ceren Akman

E-mail:ceren.akman@ahievran.edu.tr

Received Date:31/01/2022
Accepted Date:18/10/2022

Availableonline at
http://jise.btu.edu.tr/

1. Introduction

Rapid advances in technology has brought innovations to our daily life. Augmented Reality (AR) is one of the leading ones among these innovations. AR technology is based on enriching the virtual environment by adding virtual objects on existing real world assets. While Azuma [1] defines AR concept as overlapping real and virtual data, providing a simultaneous interaction between the enriched environment and the user, Milgram et al. [2] state that AR concept emerges in the part where real environment starts to become virtual. With the help of AR technology built on the real world, 2D or 3D objects created in the virtual environment can be placed on real images, thus adding the feeling of being a part of the real world to the perception of the user [3, 4].

Supporting the content with 3D data with the help of AR technology has enabled this technology to be used in many areas, especially in the field of education. Research carried out within the scope of this study shows that old-fashion teaching techniques may be insufficient for these young people due to the fact that Generation Z grows together with technology. Especially, the increase in the preference of mobile devices, such as tablets and smart phones, has provided these devices to enter the field of education and resulted in the emergence of the concept of mobile learning. Keskin et al. [5] stated that efficiency in the education process will increase thanks to the fact that students can access information practically regardless of time and place by mobile learning. Korucu et al. [6] also studied mobile learning technique and stated that positive effects occur in the field of education as the technique increases the situations such as supporting personal learning, excitement, curiosity, and the desire to research by considering the learning process with a student-centered approach. In another study, Avci et al. [7] used AR technology to teach the periodic table, which is one of the basic subjects of chemistry course. They stated that students' interest and curiosity increased by using the developed application due to the possibility of seeing and interacting all elements in 3D environment. Uzun et al. [8] developed an AR application to explain human skeletal system to children with disabilities. In the application, human anatomy and skeletal system were supported with 3D visuals, and it was observed that disabled children learned the skeletal system more easily. In the light of this finding, by examining the studies in the field of education, it can be seen that AR technology has much to add to mobile learning process.

Especially with the arrival of Generation Z, use of mobile devices, such as tablets and phones, has increased. The ease of using these devices, the possibility of performing operations with voice commands, and the increasing usage ratio among children have accelerated the integration of AR technology to mobile platforms. At the same time, having sensors such as GPS, accelerometer, and compass, mobile devices have gained many advantages in location-based AR model [9]. AR environment can easily be created by processing the data gathered from sensors. The most frequently used application in this area has been Wikitude AR [10–12].

Apart from location-based AR technology, there are AR types such as projection-based, superimization-based, and vision-based. In vision-based AR applications, images or frames of a video shot by a camera are processed. By using feature extraction methods, coordinate data of the reference object in the image is tracked, and 2D or 3D objects are virtually positioned over this reference object. Vision-based AR technology is divided into two as Marked-Based AR (MBAR) and Markerless AR (MAR).

In order to ensure real&virtual integration with MBAR applications, a marker must be introduced to the system beforehand. When the marker gets into the view of the camera of the device, application becomes active and AR

environment gets created. In the use of the MBAR model, video stream taken from the camera is separated into frame,s and a previously introduced marker is searched in each frame. The search continues till the marker gets found in a frame; at this point, a virtual object is positioned on the real-world coordinate of the marker in the frame. Finally, AR environment is created by integrating virtual reality.

When examining applications of AR technology in the field of education, we can see that MBAR model has a wide usage. However, due to its dependence on defining and using of a specific marker, it lacks potential of what?. In order to eliminate this disadvantage, MAR model, which provides environment and marker independence, can be preferred. In this model, natural targets in the image are used as reference instead of introducing a marker to the system beforehand. However, in order to create an AR environment using natural targets, feature detection, extraction and matching algorithms which require high processing power and memory capacity should be used in the background. This model is disadvantageous for mobile devices with low memory capacity and processing power. On the other hand, with proper choices on parameters, fore-mentioned algorithms can be efficiently used on mobile platforms.

Within the scope of this study, the aim was to create an AR environment by using MAR model, which provides independence from the environment and the marker. For this purpose, analyses were made on PC and mobile platforms with various combinations of feature extraction and matching algorithms on video images with different brightness, scale, and color levels. During this process, i) the number of key points in the source image, ii) the detection time of key points, iii) the number of matching key points and the required time for matching between consecutive frames, and iv) the amount of positional deviation of the repositioned virtual object depending on the key points were all examined. Performance measurements were achieved, and the most appropriate algorithm combination was selected along with proper parameters. In the second, third, and fourth sections of this study, materials and methods used in the study are clearly explained, experiments along with results are presented, and discussion on the results are given, respectively.

2. Material and Method

In order for the video frames used in the AR environment to get recognized and associated in the digital environment, there is a need to determine the descriptors, such as key points, and extract the feature vectors, which are accepted as the digital signature of these descriptors. Generally, detection operations are performed, using natural descriptors such as edges, corners, and geometric shapes in video frames. Once these descriptors are found, feature vectors, which are considered as the digital signature of the image that enables what? It is not clear. Will you please write the noun instead of the pronoun here? to be used in the virtual environment, are obtained by using feature extraction methods.

There are various methods proposed for this purpose in the literature. Within the scope of this study, Oriented FAST and Rotated BRIEF (ORB), Scale-Invariant Feature Transform (SIFT), and Speeded-Up Robust Features (SURF) key point detection and feature extraction methods were employed. Also, at the matching phase, Bruteforce, Bruteforce-Hamming, and Flannbase matchers were used to match points detected in consecutive frames. In this section, these methods are briefly mentioned.

2.1. Key point Detection and Feature Extraction

2.1.1. Scale-Invariant Feature Transform (SIFT)

SIFT, by David Lowe [13], is an algorithm that basically consists of four steps which enable detection of key points on images and extraction of corresponding feature vectors. In the use of the algorithm, first of all, a Gaussian filter is applied to different scales of the pattern and appropriate key points are determined (Figure 1) [13, 14].

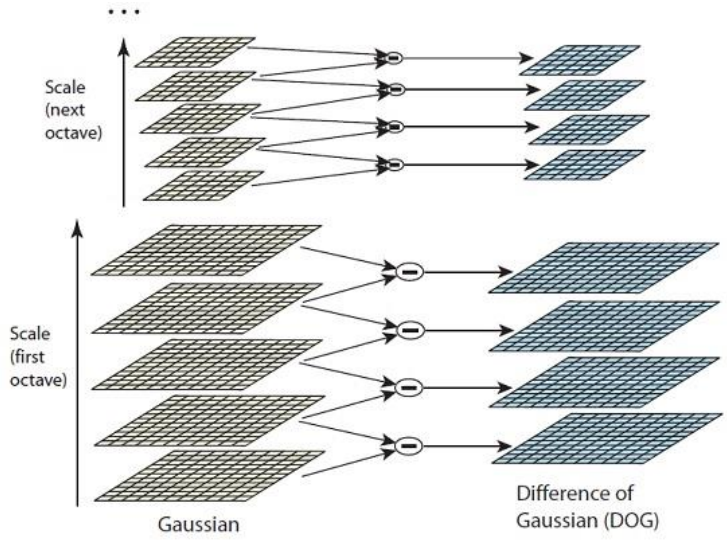


Figure 1. Using DoG at different scales [13].

Not all identified key points are stable. For this reason, locations of stable key points are determined by eliminating unstable and low-contrast points by using a quadratic Taylor series in Difference of Gaussians (DoG) space. A direction is assigned to each key point so that it is not affected by the rotation effect. In order not to get affected by other changes, a 16×16 pixel-size window is created for each key point, having the key point in the center. Then, 16 sub-windows of 4×4 size are obtained from this window; gradients are calculated for each sub-window and stored in histograms of 8 parts. By this way, 128 feature vectors having stable descriptors are extracted. SIFT algorithm is resistant to changes in rotation, scale, illumination, and perspective transformations in the image area thanks to the steps performed [13]. However, the use of this algorithm requires high computing power and memory sources. For this reason, this algorithm might be inefficient in smart mobile devices with insufficient memory capacity and processing power.

2.1.2. Speeded-Up Robust Features (SURF)

In 2006, Bay et al. [15] proposed SURF algorithm that is based on integral images and Hessian matrix as an alternative to SIFT. Similar to SIFT algorithm, SURF is resistant to changes in scale and rotation; however, it requires less computation time for key point detection and feature extraction stages as compared to SIFT. Instead of DoG method used in SIFT, box filtering method is used in SURF algorithm. In DoG method, an iterative scaling technique is used that is the result of the previous scaling is expected so that the next scaling can be performed. However, in the box filtering method, candidate key points are determined by performing different scaling operations directly on the original image instead of iterative scaling. All candidate key points are determined by using the determinant of the Hessian matrix in the current scale space, and in order to eliminate unstable candidates, the “suppress the weak one” approach is used at each scale close to 3×3×3 (Figure 2). In this approach, each

candidate key point in the center is compared with a total of 26 neighboring candidate key points in all 3 scales, and if the current value of the candidate is low, it gets eliminated [16].

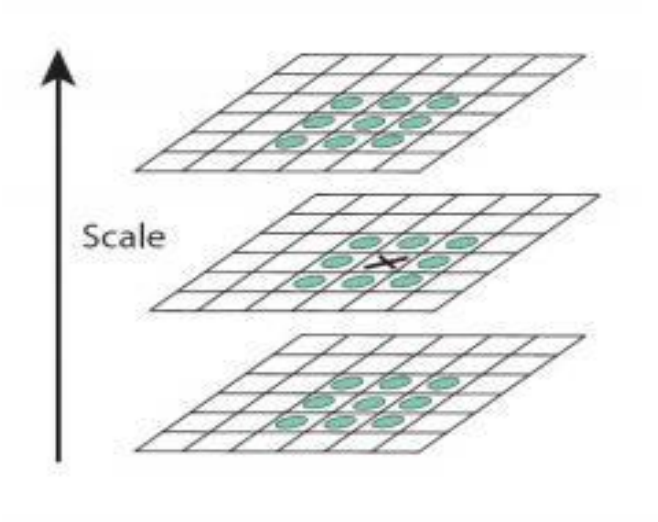


Figure 2. SURF keypoint detection [16].

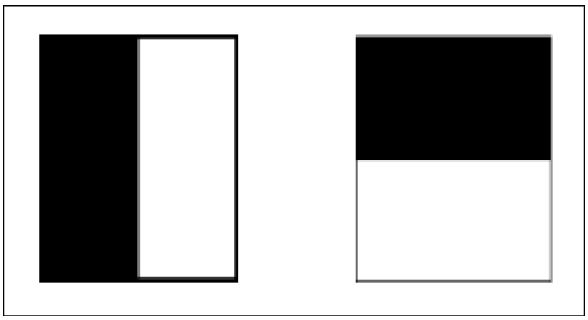


Figure 3. Haar wavelet model [16].

In SURF algorithm, a pair of Haar wavelets, shown in Figure 3, is used for x and y directions on the input image in order to increase stability of the feature vector extraction process and reduce computation time. In this model, for each key point, a neighborhood of size 20S×20S with the key point in the center is created where S is the scale level. This area is then divided into sub-regions of 4×4. By applying Haar wavelets to these sub-regions, wavelet responses dx and dy values on x and y directions are obtained, respectively. Then, in order to be sensitive to rotation, wavelet responses are weighted by 2S Gauss, and sum of the absolute values of dx and dy values are found (Figure 4). Thus, the quadruple (VSURF) shown in Equation 1 is obtained for each 4×4 sub-region. A combination of all quadruples forms the final feature descriptive vector [16, 17].

$$V_{SURF} = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right) \quad (1)$$

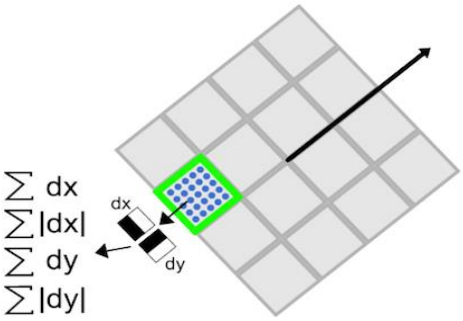


Figure 4. Descriptive component diagram [16].

2.1.3. Oriented FAST and Rotated BRIEF (ORB)

ORB is yet another key point detection and feature extraction algorithm based on FAST [18] and BRIEF [19] algorithms. It was developed by Rublee et al. [20] as an alternative to SIFT and SURF algorithms. ORB gains scale invariance by using the scale pyramid. For each scale, key points are determined by using FAST, and top N key points are quickly found by using the Harris corner metric. FAST has no orientation feature. To overcome this problem, ORB uses "Center of Gravity" method. In order to obtain the orientation vector, density centroid of the corners (C) is calculated by using Equation 2 where I is the image itself [21].

$$m_{pq} = \sum_{\forall x,y} x^p y^q I(x,y), \quad C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2}$$

After that, an $\overrightarrow{OC}$ vector is obtained from point *C* to point *O*, which is the geometric center of the vertices. Using Equation 3, direction of this vector is determined by finding the angle θ between points *O* and *C*. Thus, ORB gains the feature of orientation.

$$\theta = \arctan\left(\frac{m_{01}}{m_{10}}\right) \tag{3}$$

BRIEF, which is another component of ORB, does not provide good results in terms of rotation. To overcome this problem, a randomly chosen pixel dataset around each key point is formed. By applying an intensity-based binary test to all pixel pairs in this dataset, a new matrix P containing binary test results is obtained. Then, for orientation angle θ, rotation matrix Rθ (Equation 4) is calculated and used in Equation 5, to find the oriented matrix Pθ [19, 20].

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4}$$

$$P_\theta = R_\theta . P \tag{5}$$

2.2. Descriptor Matching

Matching phase is the next step that should be applied after key point detection and feature extraction in order to create an AR environment. In this phase, descriptor matchers such as Bruteforce, Bruteforce-Hamming, and FLANN-based matching [22, 23] are used to determine the correspondence between descriptors in different perspectives of a particular object. By this way, it is ensured that operations of great importance such as detection

and tracking of the object are carried out. The main idea behind these matchers is based on K-Nearest Neighbors algorithm which finds top K-nearest neighbors (in query image) of each descriptor from the reference image. However, the proximity values detected between two images may be erroneous due to noise. In order to prevent this error and provide a stable matching process, usually a screening system based on a predetermined threshold is used. In this system, a metric which is defined as the ratio of distance between closest pairs to distance between second closest pairs shows whether the match is correct or not. If this ratio is less than the threshold, it is considered as an acceptable match; otherwise, the next pair is evaluated.

2.3. Random Sample Consensus (RANSAC)

RANSAC algorithm was developed by Fischler and Bolles [24] to predict a strong and successful model from the dataset by eliminating outliers according to the input data. Matches found by using descriptor matching methods are not always expected to be reliable due to the noise in the image. Therefore, by using the RANSAC method, whose steps are given in Algorithm 1, this handicap can be eliminated and most accurate matches can be obtained, allowing a stable homograph estimation [25–27].

Algorithm 1: Steps of RANSAC algorithm.

1- S samples are randomly chosen from the dataset X consisting of correct matches, and the model parameter is determined according to selected samples.
2- Points within a specified threshold value t are assigned to the model parameter and the dataset X_i is determined according to these assigned values. The dataset X_i is the consensus of the sample data.
3- If the size of X_i is greater than threshold t, then, the model is re-estimated by using all points in X_i and the process is finished.
4- If the size of X_i is less than threshold t, again, S samples are randomly chosen from dataset X and previous steps are repeated.
5- After N trials, the largest consensus set X_i is selected, and the model is recreated by using all points in the X_i dataset.

3. Results and Discussion

Numerical data associated with the figures in this section can be found at https://github.com/akmancrn/article-graphics-results.

AR assisted education is one of the common application fields of AR technology and is chosen as the subject of this study. By the literature review, it was seen that majority of the applications developed for this area is based on MBAR, which relies on a predefined specific marker. However, relying on such a marker restricts the practicality of applications. Because of this disadvantage, MAR, which accepts natural features in the view as reference and provides environment and marker independence, is studied instead.

MAR model, as mentioned in the previous sections, is composed of key point detection, feature extraction, and matching processes. These processes of MAR use algorithms which require high computation power and memory resource. In order to compensate these needs and create a stable AR environment, tests were performed on both PC

and mobile platforms with reasonable sources. PC platform had a 6-cores CPU running at 4.50GHz, and 16GB RAM while the mobile platform had a 4-cores CPU running at 2.0GHz, and 2GB RAM.

In order to create an AR environment on mentioned platforms, a 433-frames long video with the resolution of 1280×800 pixels were shot using the mobile platform. The first frame was selected as the reference image, and the remaining 432 frames were accepted as query images. These images were processed with key point detection and feature extraction algorithms such as ORB, SIFT, and SURF under different brightness (Figure 5), and scaling conditions, both on mobile and PC platforms. After that, the performance of these algorithms was compared by using test results obtained from all query images.

(a) Normal brightness level. (b) 100% increased brightness level.

Figure 5. Different brightness levels of the sample video.

First of all, average runtimes on PC platform (average of runs at 10 different times) for different scaling levels of the query images were examined. In the examination, current brightness level was kept constant (normal brightness).

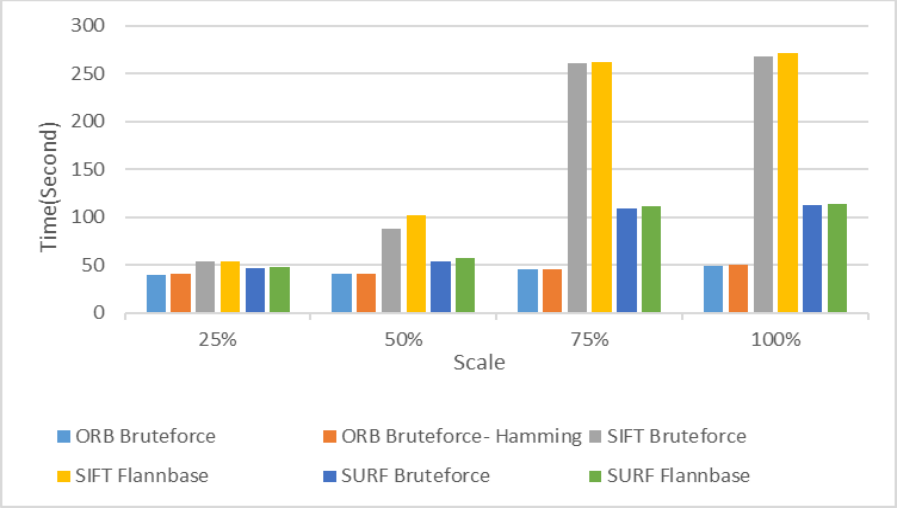


Figure 6. Average runtimes at normal brightness level on PC platform.

Figure 6 shows runtimes of various combinations of detection, extraction, and matching algorithms on PC platform for four different scaling levels when the brightness is kept at normal level. It can be clearly seen that ORB-based combinations were the fastest and SIFT-based combinations were the slowest ones in terms of running time.

When the brightness level was increased by 100%, average runtimes (average of 10 different runs) of various combinations of detection, extraction, and matching algorithms on PC platform for four different scaling levels were found as shown in Figure 7.

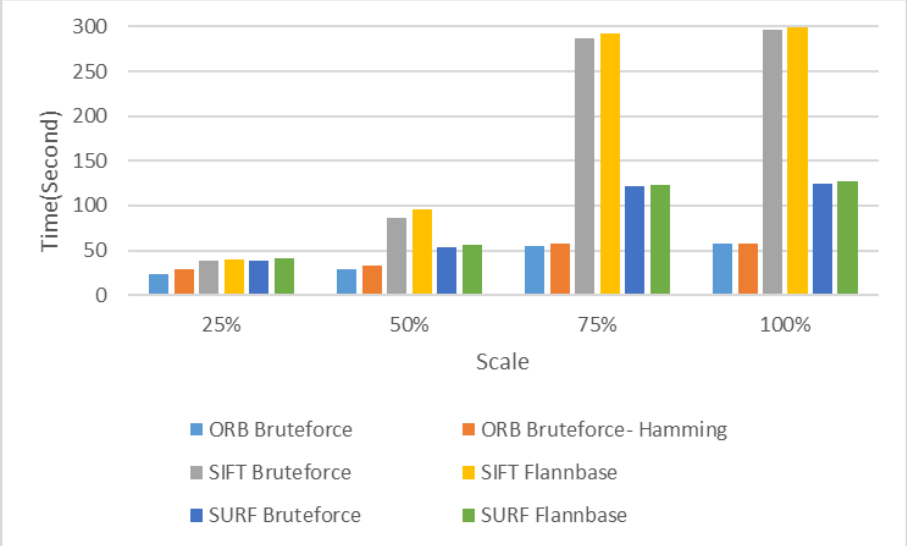


Figure 7. Average runtimes at 100% increased brightness level on PC platform.

As seen in Figure 7 as well, ORB-based combinations were the fastest and SIFT-based combinations were the slowest ones when there was a 100% increase in brightness level. Compared to Figure 6, a negligible increase in average runtimes was observed when the brightness level was increased.

In a similar fashion, Figure 8 and 9 show average runtimes of combinations of detection, extraction, and matching algorithms on mobile platform, under normal and 100% increased brightness levels, respectively.

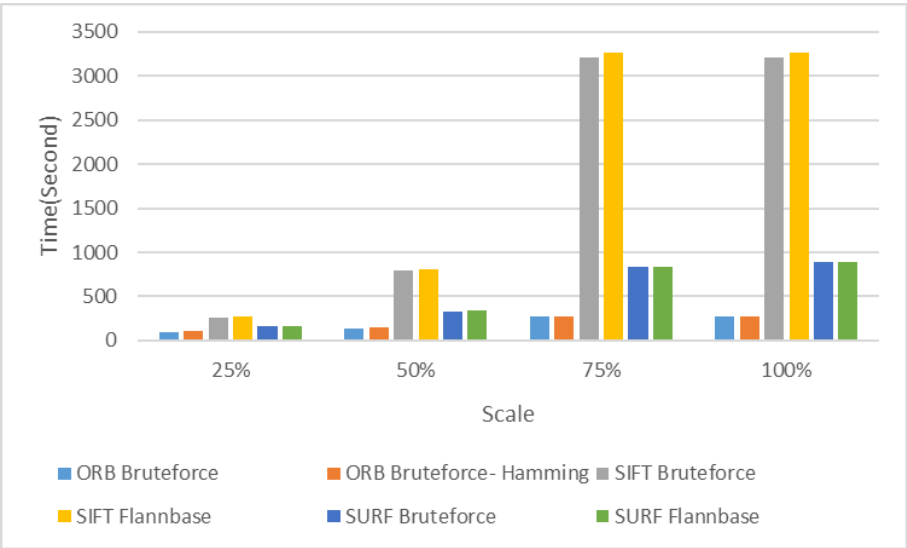


Figure 8. Average runtimes at normal brightness level on mobile platform.

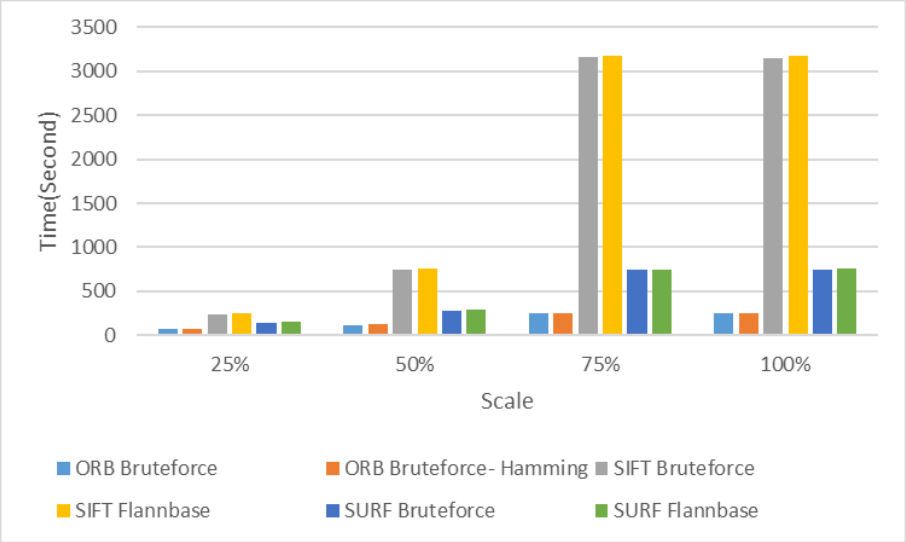


Figure 9. Average runtimes at 100% increased brightness level on mobile platform.

As similar to the previous results, ORB-based combinations were the fastest and SIFT-based combinations were the slowest ones under different brightness levels on mobile platform. However, considering average runtimes, it was observed that the time spent on mobile platform was much higher than the time spent on PC platform. Considering the figures, it is obvious that runtimes can be reduced by decreasing the scaling level, which is an advantage for mobile platforms.

Key point detection and feature extraction processes directly affect average runtime (average of 10 different runs). For normal brightness (NP) and increased brightness (AP) levels, average detection times of all key points on the PC platform are given in Figure 10.

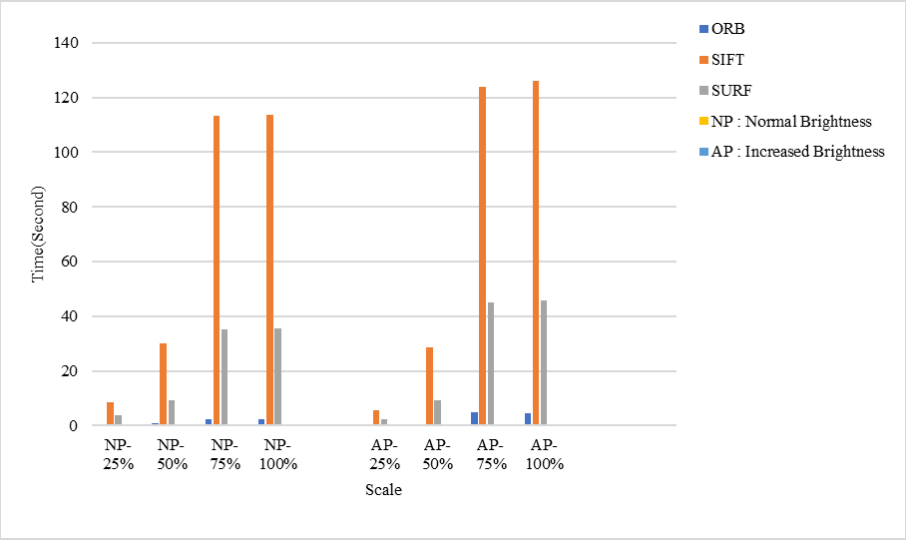


Figure 10. Average keypoint detection times under NP/AP conditions on PC platform.

When average key point detection times for both platforms were examined, it was observed that the process was completed at the slowest pace by using SIFT and fastest using ORB. Test runs were also performed on mobile platform. It was observed that the application on mobile platform runs 13 times slower than the one on PC platform. However, ratios of runtimes of algorithms remained the same.

Another factor affecting the total runtime is feature extraction process. For PC and mobile platforms, average times to extract the feature vector at normal brightness and increased brightness levels are given in Figure 11 and Figure 12, respectively. Examining the figures, it was observed that feature extraction process was again completed,at the slowest pace by using SIFT and at the fastest pace by using ORB.

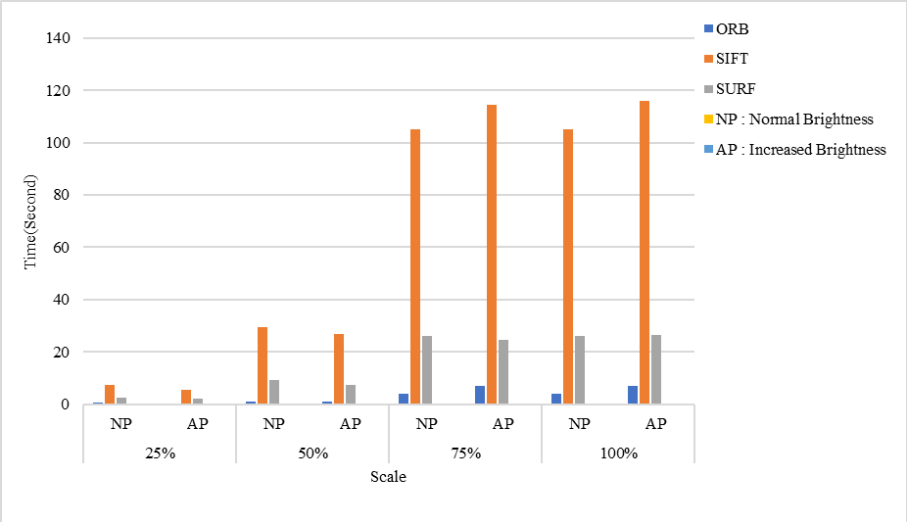


Figure 11. Feature extraction times under NP/AP conditions on PC platform.

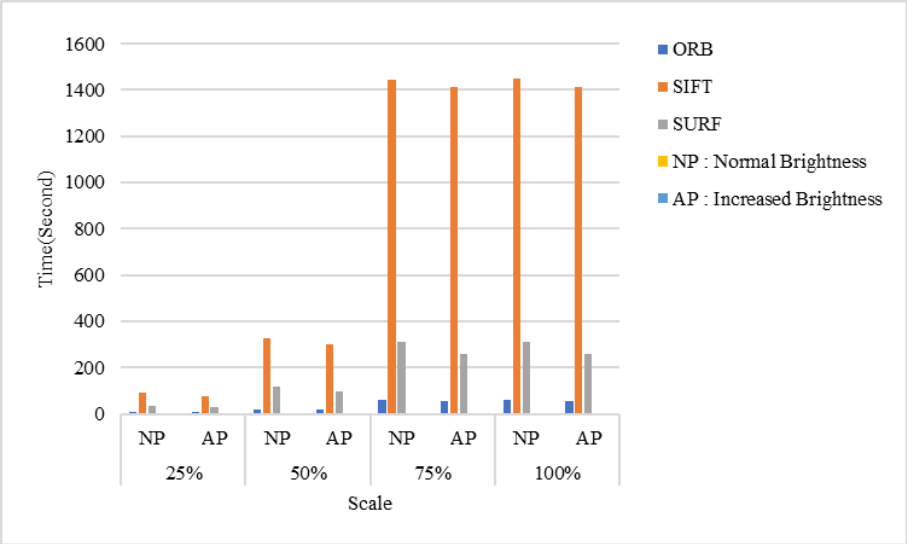


Figure 12. Feature extraction times under NP/AP conditions on mobile platform.

Number of key points obtained from all query images at normal brightness level was the same for both platforms,as shown in Figure 13. The same conclusion can be made for the increased brightness level, the results of which are given in Figure 14. In terms of the number of key points, it was seen that SURF detected the most while ORB detected the least. Figure 10 reveals that, SIFT was the slowest algorithm, while SURF detected more key points than any other algorithms. This shows that box filtering approach used in SURF is more advantageous than Gaussian filtering approach used in SIFT. Similar results have been reported in studies in the literature [29-30].

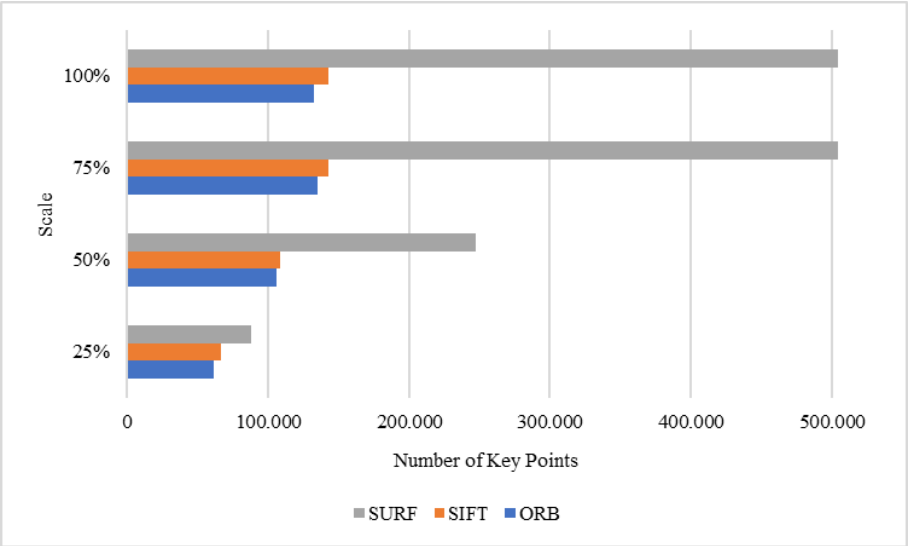


Figure 13. Number of key points at normal brightness level on PC/mobile platform.

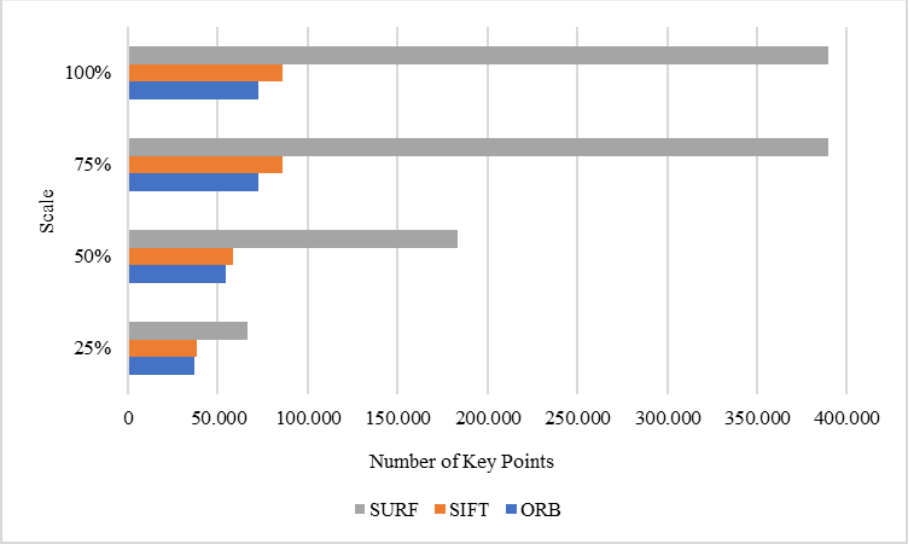
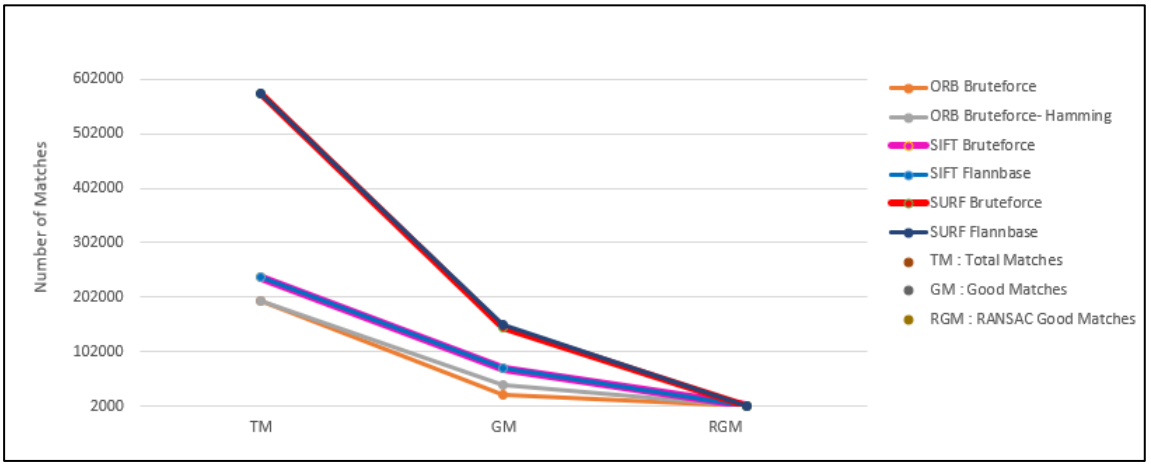


Figure 14. Number of key points at increased brightness level on PC/mobile platform.

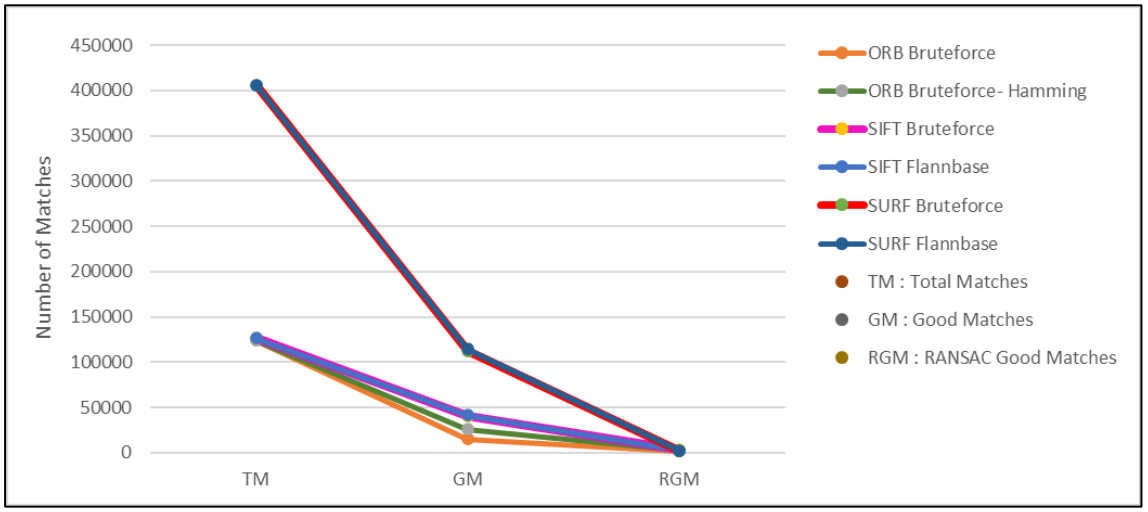
Increasing the brightness level at all scaling levels for mobile platform and at 25%, 50% scaling levels for PC platform reduced the average runtime. However, increasing the brightness level at 75%, 100% scaling levels on PC platform also resulted in an increase in the average runtime (Figure 7). While the increase and decrease in runtimes were negligible, it is not possible to reach the same conclusion for the number of key points. When the brightness level was increased, the number of detected key points decreased to a level that could not be neglected (Figure 13 vs. Figure 14). Thus, it can be seen that, although increasing the brightness level may seem advantageous in terms of operating time, this may be a disadvantage in terms of the detected key points, causing loss of valuable data. This is a trade of for mobile platforms.

Following the key point detection and feature extraction steps, matching methods are used for scene tracking. For this step, Bruteforce, Bruteforce-Hamming, and Flannbase matchers were used together with K-Nearest Neighbors (KNN) algorithm. Using this algorithm, each descriptor (Q) in the query image is matched with the two nearest neighbor descriptors (R1 and R2) in the reference image. By this way, images obtained from the sample video were

compared with the reference image, and the matching process was completed. However, detected matches were not entirely correct. Therefore, the distance between matching descriptors was compared to a preselected threshold value. Considering the present studies in literature, this value is generally chosen as 0.8 [28]. In the calculation process, if the distance between Q and R1 was less than threshold × the distance between Q and R2, then, Q-R1 pair was accepted as a correct matching pair represented by GM (Good Matches). However, all GM were still not reliable due to the noise in images. At this point, in order to overcome the problem, researchers have shown that absolute correct matches can be detected by using RANSAC algorithm [29-30]. Thus, in this study, RANSAC algorithm was used to make a more accurate homograph estimation. The most accurate matching pairs obtained by using RANSAC are expressed by RGM (RANSAC Good Matches). For both brightness levels,the number of matches obtained in all cases (TM: Total matches, GM: Good Matches, RGM: RANSAC based Good Matches) are given in Figures 15, 16, and 17. Figure 15 shows results for 100% and 75% scaling levels, Figure 16 shows results for 50% scaling level, and Figure 17 shows results for 25% scaling level.

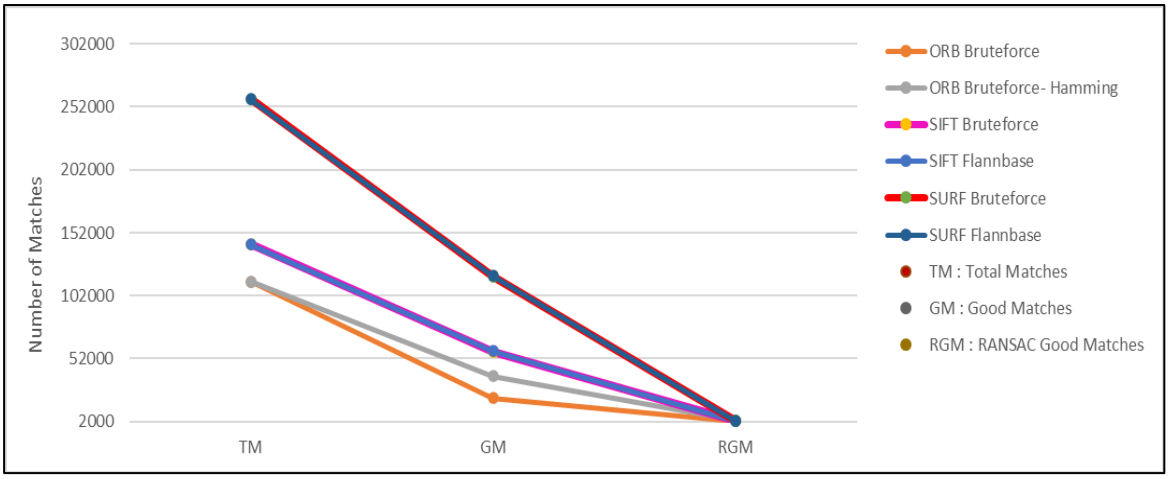


(a) Normal brightness.

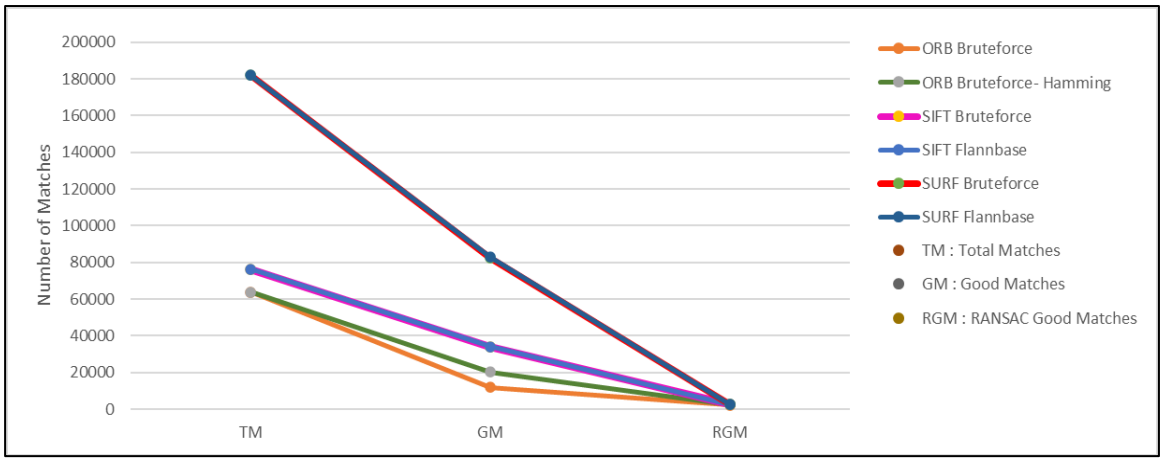


(b) Increased brightness.

Figure 15. Number of matches for 100% and 75% scales on both platforms.
(a): Normal brightness, (b): Increased brightness

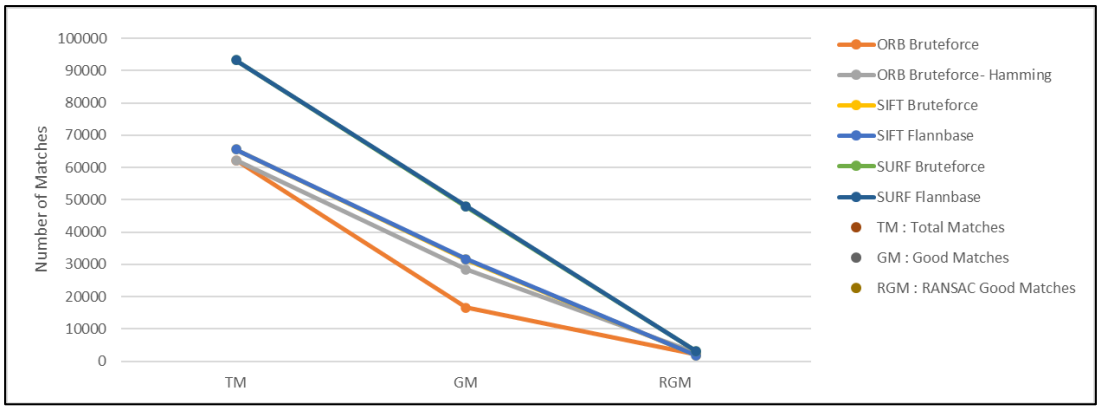


(a) Normal brightness.

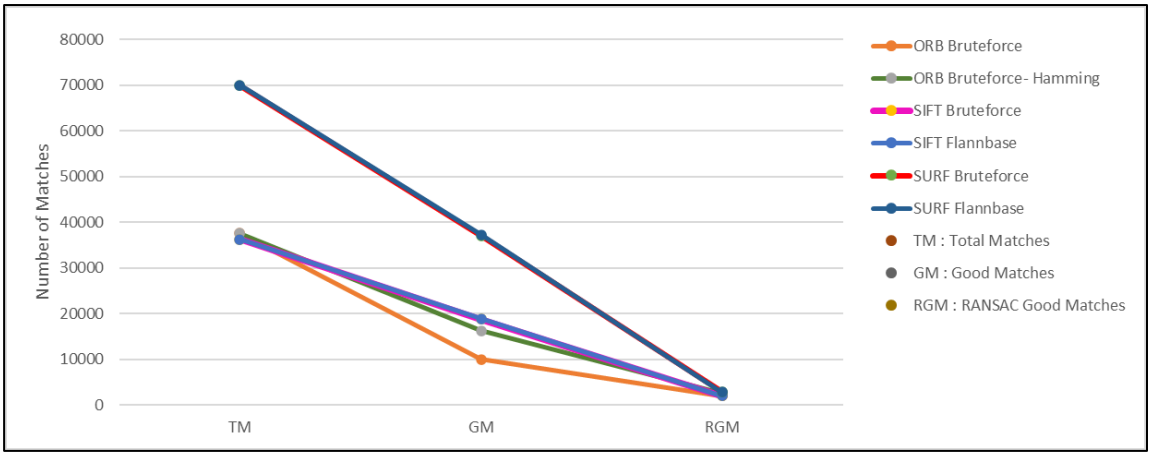


(b) Increased brightness.

Figure 16. Number of matches for 50% scale on both platforms.
(a): Normal brightness, (b): Increased brightness



(a) Normal brightness.



(b) Increased brightness.

Figure 17. Number of matches for 25% scale on both platforms.
(a): Normal brightness, (b): Increased brightness

Considering the number of total matches, matching after SURF gives the highest number of matches while matching after ORB gives the lowest. By applying Equations 6 and 7, proportional GM% and proportional RGM% rates were also obtained (Figure 18), respectively. These values are the same for both PC and mobile platforms. It was observed that GM% was the highest for SIFT-derived descriptors at 100% and 75% scaling levels, and for SURF-derived descriptors at 50% and 25% scaling levels. However, when examining RGM%, it was observed that RGM% was the highest for ORB-derived descriptors at all scaling levels. This reveals that RANSAC algorithm chooses more descriptors derived by ORB than SIFT or SURF. GM% and RGM% rates for the increased brightness level are also given in Figure 19.

$$GM\% = GM/TM * 100 \quad (6)$$

$$RGM\% = RGM/GM * 100 \quad (7)$$

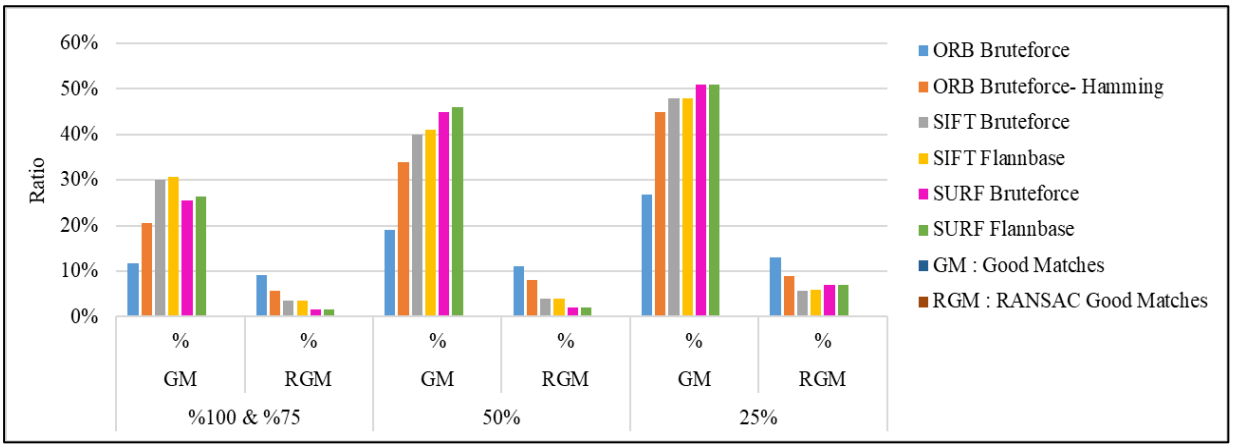


Figure 18. Matching rates at normal brightness level.

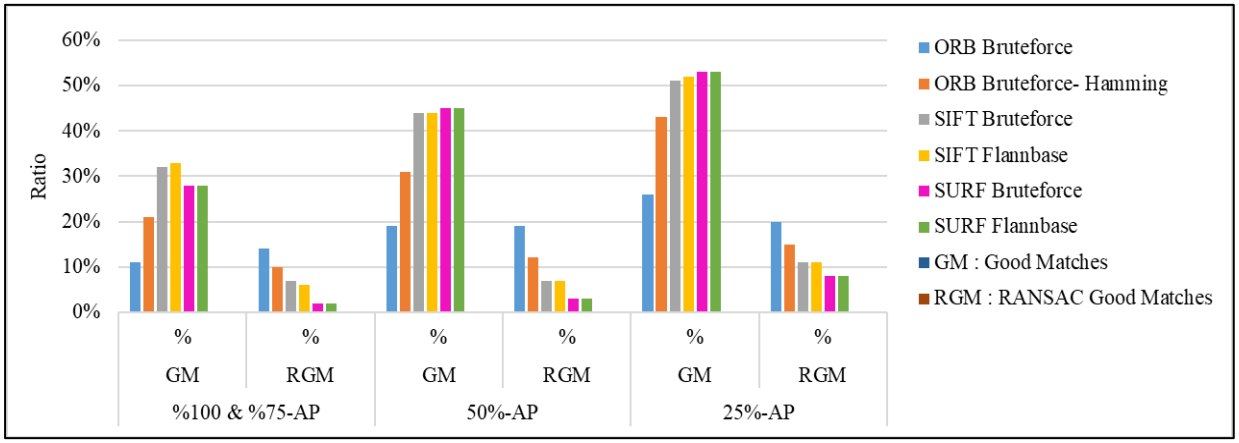


Figure 19.. Matching rates at increased brightness level.

Another subject of the study is to compare the performance of detection&matching combinations in terms of positioning a virtual object. For this purpose, a virtual object was placed in the center of the black circle shown in Figure 5 which is the first frame (reference image) of the video. Then, this object was repositioned on following 432 frames (query images) of the video by using various descriptor&matcher combinations such as OB (ORB Bruteforce), OBH (ORB Bruteforce-Hamming), SB (SIFT Bruteforce), SFL (SIFT Flannbase), SFFL (SURF Flannbase), and SFB (SURF Bruteforce). At each repositioning step, one might expect to see the virtual object maintaining its position in the very center of the black circle; however, this does not happen all the time causing a deviation. In order to measure the amount of this deviation, the distance between the new position of the object and the center of the circle should be calculated at each repositioning step. For this, the center of the circle in each image was determined by using HoughCircle method provided by OpenCV 2.4.11 library. The amount of deviation changes depending on the used descriptor&matcher combinations, brightness level, and scaling level (Figure 20).

	Scale Size			
	25%	50%	75%	100%
δ	320*200	640*400	960*600	1280*800

Figure 20. δ value for each scaling level.

By using Equations 8 and 9, pixel-based deviation value (DV) and deviation ratio subject to the image scale (DRS) were calculated for each query image, respectively. Afterwards, a total of 10 query images with the lowest and highest deviations were determined as outliers and got eliminated. Then, a mean deviation ratio (MDR) over all query images was calculated.

$$DV = \frac{\sqrt{x^2+y^2}}{\delta} \tag{8}$$

$$DRS = DV * 100 \tag{9}$$

The mean deviation ratios at each scaling level were examined separately for both platforms, at normal brightness level, and without using RANSAC. For 100% scaling level, it was observed that the highest MDR occurred by using SIFT-based descriptor&matcher combinations, while the lowest MDR occurred by using ORB-based descriptor&matcher combinations (Figure 21).

Matching rates and the number of key points are numerically the same for 75% and 100% scaling levels (see Figures 13-14, 18-19). However, the resolution obtained as a result of scaling on the image is different. Therefore, MDR differs for these two scaling levels. Examining Figure 21, it can be said that the decrease in the scaling level reduced the deviation ratio for SURF-based descriptor&matchers while this is not valid for other combinations. On the other hand, when the deviation ratio axis is examined, one can see that there are deviations above 100%. This indicates that the virtual object deviates too much or even goes off the screen.

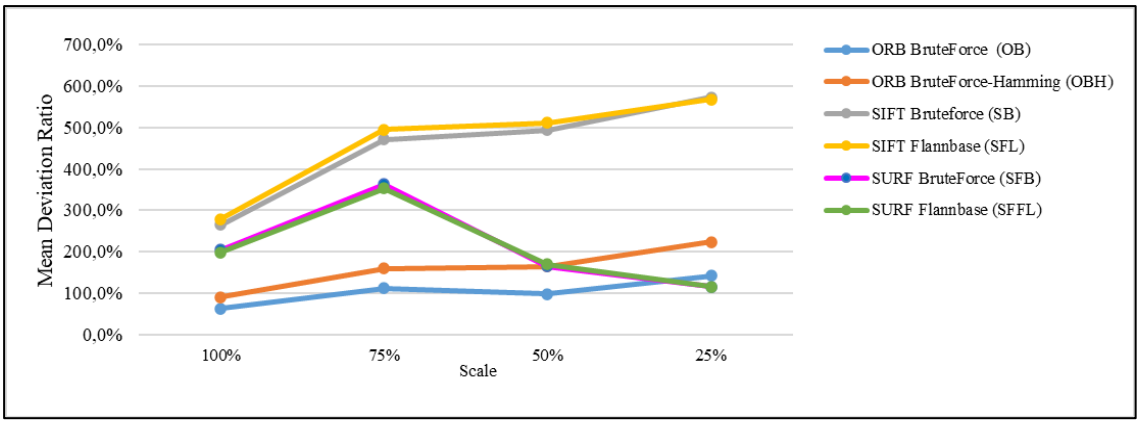


Figure 21. MDR without RANSAC at normal brightness level on both platforms.

MDR values in Figure 22 were obtained by using RANSAC algorithm at normal brightness level for both platforms. Paying attention to the deviation ratio axis, one can see that the virtual object remained on the screen for all descriptor&matcher combinations at all scaling levels, even if it deviated from the position it should be. It is observed that, for all combinations, there is less deviation when using the original 100% scaling level and more deviation at 25% scaling level.

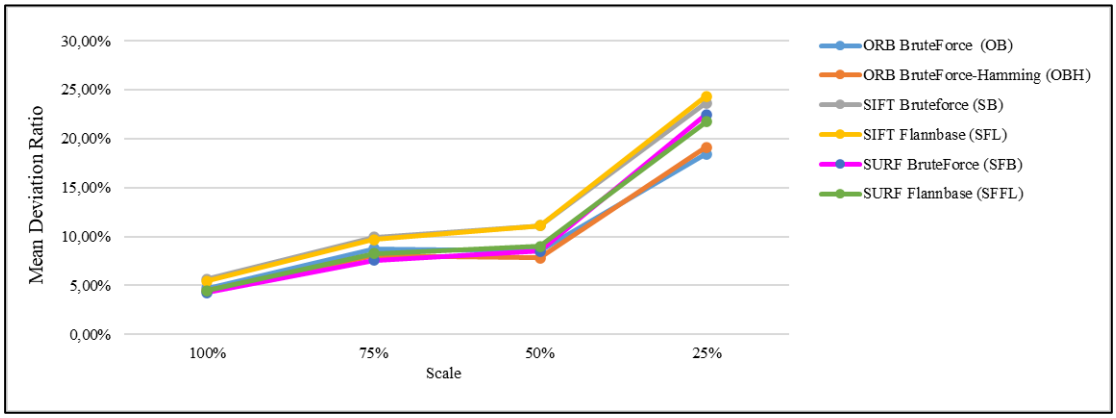


Figure 22. MDR with RANSAC at normal brightness level on both platforms.

Similar tests were done, using the test video with the increased brightness. When the deviation ratio axis in Figure 23 is examined, without using RANSAC, MDR value is observed to be less than MDR value obtained by using the normal brightness level. However, MDR values are mostly above 100%. This indicates that the virtual object usually moves on screen and is mostly off-screen. When deviations of all combinations at various scaling levels were examined, it was observed that the least deviation occurred in OOB combination at 100% scaling level, and the highest deviation occurred in SFSFL combination at 75% scaling level.

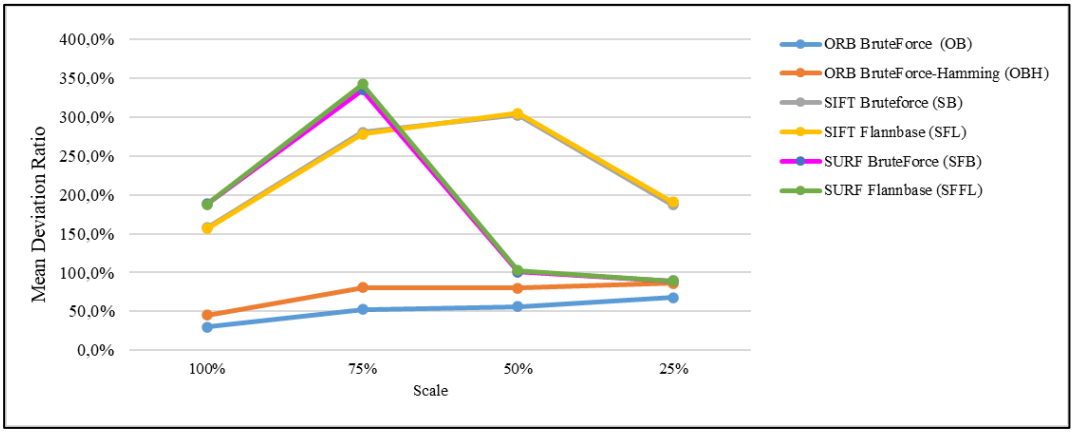


Figure 23. MDR without RANSAC at increased brightness level on both platforms.

MDR values in Figure 24 were obtained by using RANSAC algorithm at increased brightness level for both platforms. When the deviation ratio axis is examined for all combinations, the virtual object is seen to deviate from its intended position but remains on the screen at all scaling levels. It is also observed that the virtual object deviates the least at 100% scaling by using OOB combination, and the most at 25% scaling level by using SSFL combination. RANSAC algorithm seems to be effective in reducing the deviation and gives more accurate results as resolution approaches to its original scale level.

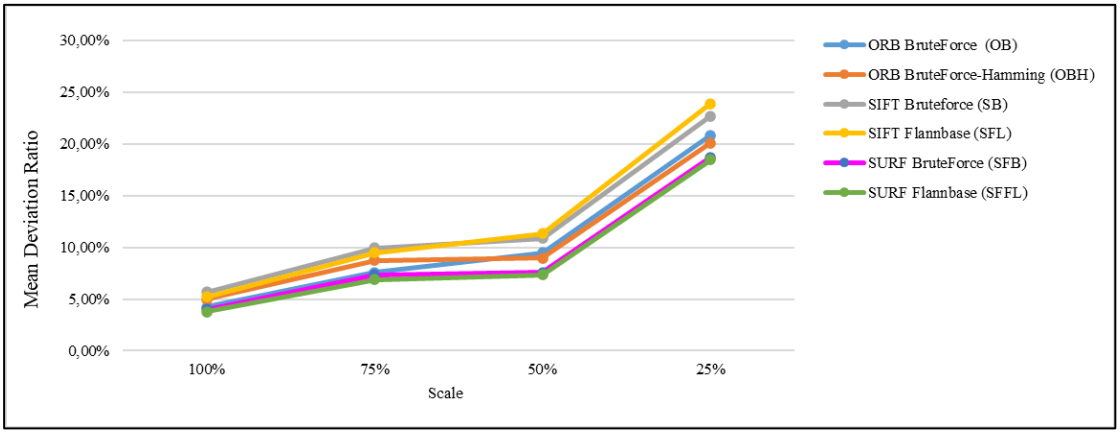


Figure 24. MDR with RANSAC at increased brightness level on both platforms.

4. Conclusion

The comparative analysis of the feature extractors used in this study is compatible with the results of previous studies [30-31]. A comparative performance analysis of the combinations of various key point detectors, feature extractors, and matcher algorithms at different brightness/scale levels has been made on mobile and PC platforms. By using these combinations, key points and corresponding feature vectors in frames of a test video were extracted and matched. The time spent extracting and matching the key points, the number of key points detected, the number of matches and the amount of deviation of the virtual object added to the image are the main areas of interest of the study.

In key point detection and feature extraction algorithms used in the study, according to the hardware used,

brightness level and scale levels, gave various results in outputs such as the number of key points, runtime, and match rate. When all the results are examined, the increase in the brightness level is observed to have a reducing effect on the number of key points detected and the key point extraction time, but it has no significant effect on the amount of deviation. This shows that the increase in the brightness level naturally leads to elimination of weak key points. This can be interpreted as an advantage for mobile platforms with low processing power.

Rather than the reduction in runtime, the most important issue is the match rate in the output, so RANSAC algorithm was used to improve stability. With the use of the algorithm, mean deviation ratio was reduced to approximately 5% for all descriptor&matcher combinations at 100% scaling level. However, when the scale level was reduced to a quarter, it was observed that the mean deviation ratio increased approximately four folds. This shows that a MAR model that makes use of the aforementioned algorithms will be significantly affected by the image scale. At this point, especially for mobile platforms, the use of ORB-based detector & matcher combination with RANSAC may be a suitable choice due to the advantages in runtime, key point matching rate, and low deviation ratio.

As future work, it is aimed to hybridize the current study with deep learning networks used for object recognition.

References

[1] Azuma, R. T. (1997). A Survey Of Augmented Reality. Presence: Teleoperators & Virtual Environments, 6(4), 355-385.

[2] Milgram, P., & Kishino, F. (1994). A Taxonomy Of Mixed Reality Visual Displays.Ieice Transactions On Information And Systems, 77(12), 1321-1329.

[3] Ufkes, A., & Fiala, M. (2013, May). A Markerless Augmented Reality System For Mobile Devices. In 2013 International Conference On Computer And Robot Vision (pp. 226-233). IEEE.

[4] Gonzato, J. C., Arcila, T., & Crespin, B. (2008). Virtual Objects On Real Oceans. In Graphicon'2008 (Pp. 49-54).

[5] Keskin, N. Ö., & Kilinç, A. G. H. (2015). Mobil Öğrenme Uygulamalarına Yönelik Geliştirme Platformlarının Karşılaştırılması ve Örnek Uygulamalar. Açıköğretim Uygulamaları Ve Araştırmaları Dergisi, 1(3), 68-90.

[6] Korucu, A. T., & Biçer, H. (2019). Mobil Öğrenme: 2010-2017 Çalışmalarına Yönelik Bir İçerik Analizi. Trakya Eğitim Dergisi, 9(1), 32-43.

[7] Avcı, A. F., & Taşdemir, Ş. (2019). Artırılmış Ve Sanal Gerçeklik ile Periyodik Cetvel Öğretimi. Selcuk University Journal Of Engineering Sciences, 18(2), 68-83.

[8] Uzun, Y., Bilban, M., & Kalaç, M. Ö. (2018). Artırılmış Gerçeklik Kullanılarak Engelli Çocukların Öğrenme Yeteneklerinin Geliştirilmesi. Uluslararası Engelsiz Bilişim Kongresi.

[9] Schall, G., Grabner, H., Grabner, M., Wohlhart, P., Schmalstieg, D., & Bischof, H. (2008, June). 3d Tracking In Unknown Environments Using On-Line Keypoint Learning For Mobile Augmented Reality. In 2008 IEEE Computer Society Conference On Computer Vision And Pattern Recognition Workshops (Pp. 1-8). IEEE.

[10] Kağan, G. Ü. L., & Şahin, S. (2017). Bilgisayar Donanım Öğretimi İçin Artırılmış Gerçeklik Materyalinin Geliştirilmesi ve Etkililiğinin Incelenmesi. Bilişim Teknolojileri Dergisi, 10(4), 353-362.

[11] Içten, T., & Güngör, B. A. L. (2017). Artırılmış Gerçeklik Üzerine Son Gelişmelerin ve Uygulamaların Incelenmesi. Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji, 5(2), 111-136.

[12] Kaleci, D., Demirel, T., & Akkuş, I. (2016). Örnek Bir Artırılmış Gerçeklik Uygulaması Tasarımı. xviii.

Akademik Bilişim Konferansı, Aydın, Türkiye.

[13] Lowe, D. G. (2004). Distinctive Image Features From Scale-Invariant Keypoints. International Journal Of Computer Vision, 60(2), 91-110.

[14] https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html [Accessed: April 4th, 2021].

[15] Bay, H., & Tuytelaars, T. (2006). Luc Van Gool. Surf: Speeded Up Robust Features, 14.

[16] Evans, C. (2009). Notes On The Opensurf Library. University Of Bristol. Tech. Rep.

[17] Li, A., Jiang, W., Yuan, W., Dai, D., Zhang, S., & Wei, Z. (2017). An Improved Fast+ Surf Fast Matching Algorithm. Procedia Computer Science, 107, 306-312.

[18] Viswanathan, D. G. (2011). Features From Accelerated Segment Test (Fast) Deepak Geetha Viswanathan 1.

[19] Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010, September). Brief: Binary Robust Independent Elementary Features. In European Conference On Computer Vision (Pp. 778-792). Springer, Berlin, Heidelberg.

[20] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). Orb: An Efficient Alternative To Sift Or Surf. In 2011 International Conference On Computer Vision (Pp. 2564-2571). IEEE.

[21] Rosin, P. L. (1999). Measuring Corner Properties. Computer Vision And Image Understanding, 73(2), 291-307.

[22] Muja, M. (2009). Approximate Nearest Neighbors With Automatic Algorithm Configuration. In Proc. Int. Conf. On Computer Vision Theory And Applications, Lisbon, 2009.

[23] https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html [Accessed: Augustth 29, 2021].

[24] Fischler, M. A., & Bolles, R. C. (1981). Random Sample Consensus: A Paradigm For Model Fitting With Applications To Image Analysis And Automated Cartography. Communications Of The Acm, 24(6), 381-395.

[25] Dihkan, M. (2019). Uzaktan Algılanmış Görüntülerin Surf Özellik Verileri Ve Ransac Algoritması Ile Otomatik Çakıştırılması. Gümüşhane Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 9(3), 425-432.

[26] Erdöl, E. S. (2019). Anahtar Noktası Ve Yama Eşleşme Yöntemleri Ile Doku Bazlı Görüntü Sahteciliği Tespiti (Doctoral Dissertation, Karadeniz Teknik Üniversitesi).

[27] Çömert, R., & Avdan, U. Yersel Lazer Tarayici Verilerinden Basit Geometrik Yüzeylerin Otomatik Olarak Çıkarılması.

[28] A. Jakubović and J. Velagić, "Image Feature Matching and Object Detection Using Brute-Force Matchers," 2018 International Symposium ELMAR, 2018, pp. 83-86, doi: 10.23919/ELMAR.2018.8534641.

[29] Li, H., Qin, J., Xiang, X., Pan, L., Ma, W., & Xiong, N. N. (2018). An efficient image matching algorithm based on adaptive threshold and RANSAC. IEEE Access, 6, 66963-66971.

[30] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2018, pp. 1-10, doi: 10.1109/ICOMET.2018.8346440.

[31] Karami, E., Prasad, S., & Shehata, M. (2017). Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. arXiv preprint arXiv:1710.02726.