International Journal on Advanced Science Engineering Information Technology

Vol.8 (2018) No. 4-2
ISSN: 2088-5334

An Efficient and Robust Mobile Augmented Reality Application

Siok Yee Tan[#], Haslina Arshad[#], Azizi Abdullah[#]

[#] *Center for Artificial Intelligence and Technology, Faculty of Information Science and Technology,*
Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor Darul Ehsan, Malaysia.
E-mail: esther@ukm.edu.my, haslinarshad@ukm.edu.my, azizia@ukm.edu.my

Abstract*— Augmented Reality (AR) technology is perceived to be evolved from the foundation of Virtual Reality (VR) technology. The final objective of AR is to offer ubiquitous access and better management to information through the use of seamless techniques in which the interactive computer-generated world is combined with the interactive real world in a coherent environment. The direction of research in the field of AR has been shifted from traditional Desktop based mediums to the mobile devices such as the smartphones. However, image recognition on smartphones executes many challenges and restrictions in the form of robustness and efficiency which are the general performance measurement of image recognition. Smart phones have limited processing capabilities as compared to the PC platform, hence the process of mobile AR application development and use of image recognition algorithm need to be emphasised. The processes of mobile AR application development include detection, description and matching. All the processes and algorithms need to be properly selected in order to create a robust and efficient mobile AR application. The algorithm used in this work for detection, description and matching are AGAST, FREAK and Hamming distance respectively. The computation time, robustness towards rotation, scale and brightness are evaluated. The dataset used to evaluate the mobile AR application is the benchmark dataset; Mikolajczyk. The results showed that the mobile AR application is efficient with a computation time of 29.1ms. The robustness towards scale, rotation and brightness changes of the mobile AR application also obtained high accuracy which is 89.76%, 87.71% and 83.87% respectively. Hence, combination of algorithm AGAST, FREAK and Hamming distance are suitable to create an efficient and robust mobile AR application.

***Keywords*— augmented reality; image recognition algorithm; detection; description; matching;**

I. INTRODUCTION

Augmented Reality (AR) is the combination of virtual content and the real world where it allows users to interact with virtual objects in real world and real time [1]. The extensive use of AR in mobile devices such as tablets and smartphones has become a growing phenomenon due to the increasing popularity of mobile devices worldwide [2]–[5]. Mobile AR allow users to enjoy the combination of virtual and real world experiences in the palm of their hands [6]. There are mainly two reasons for its success; the hardware and the available tools for programming. Tablets and smartphones nowadays are designed for a wide range of consumers with unexpected robust for an AR application performance [7], [8]. Most tablets and smartphones come with a built-in camera which allows the computer vision approaches like AR to be used by a modern-day consumer. However, the process of developing mobile AR and choosing the right image recognition algorithm need to be underlined in order to achieve optimum performance for mobile AR application as smart phone has limited processing capabilities as compared to the PC platform [9].

AR application may during the application process. It may occur due to number of reasons for example, sudden changes in distance of the marker and the camera or sudden changes in lighting [10]. It is important that the application can recover from it as smoothly as possible after a failure. The process of mobile AR application can be improved by implementing effective and robust image recognition algorithm. The image recognition process to develop a mobile AR application involve the following process; image detection, image description and image matching. Hence, the main target of this research is to present the implementation of an image recognition algorithm in producing a high performance mobile application. The detector, descriptor and matcher used in this resesarch are AGAST, FREAK and Hamming distance respectively.

The rest of the paper is organised as follows. After a discussion on the material and method in Section II, result and discussion are discussed in Section III and the conclusion are discussed in Section IV.

II. Material and Method

A. Image Recognition Algorithm

Image recognition in mobile AR can be divided into three main components; detector, descriptor and matcher. Basically, an image will be captured through video frame by using phone's camera and the image is converted into grey scale image. The first step in image recognition process is image feature detection where feature detectors were used to identify the keypoints or the natural features of an image. The next step is to obtain a descriptor of each keypoints from the image. A feature descriptor is required to describe or extract the keypoints that had been detected or extracted in the first step; detection process. Descriptors can be categorised into two categories; floating-point descriptors and binary descriptors. SURF (Speeded-Up Robust Feature) and SIFT (Scale Invariant Feature Transform) are good instances of floating-point descriptors, while FREAK (Fast Retina Keypoint), BRISK (Binary Robust Invariant Scalable Keypoints), ORB (Oriented Fast and Rotated BRIEF) and BRIEF (Binary Robust Independent Elementary Feature) are another instances of binary descriptors. Matching the feature descriptor is the third step in the image recognition process. The features or keypoints of the reference image should be kept first in the database to allow the application to match the points of a query image with those of reference image during the matching process [11].

1) Feature Detection: Detecting or identifying features used for image recognition can be traced a long way back in the literature. Harris & Stephens [12] proposed one of the earliest and probably most famous detectors; corner and edge detector. Mikolajczyk [13] introduced a comprehensive assessment of the most efficient detection method at the time, indicating no single versatile detector unless the additional properties of the different approaches depend on the context of the application. Another recent feature detector, FAST (Features from Accelerated Segment Testis) is based on a characteristic feature criterion accelerated segment test (AST) [14]. FAST is an order of magnitude faster than other feature detectors such as Difference of Gaussians (DoG) used by SIFT [15] and SURF [16]. FAST detector is used in real time application which have limited computational resources. FAST has obvious advantages from many perspective. However, there is still imperfection in limiting FAST to be used in computer vision applications. In order to improve the processing speed, FAST is enhanced by the machine learning algorithm ID3 [17]. This machine learning algorithm is a way to yield a decision tree from a training dataset. FAST needs to be trained on an image dataset from its work environment, and then a decision tree is obtained to verify each center pixel whether it can be a feature or not. However, this approach does not guarantee that each pixels combination can be found; this may produce erroneous results. In addition, the FAST detector must be trained each time when the working environment changes. This limits FAST to work AR application that work without any prior environmental knowledge [18].

In order to overcome the weakness of FAST, AGAST (Adaptive and Generic Corner Detection Based on the Accelerated Segment Test) feature detector is proposed [19]. AGAST followed the same AST feature criterion similar to FAST but AGAST uses a different decision tree. AGAST is trained based on a unique dataset. This dataset includes all possible combinations of 16 pixels on the circle. This assure that the decision tree is suitable for any work environments. This makes AGAST ideal for real-time computer vision applications such as mobile AR application. Therefore, AGAST had been selected as feature detector to detect image features in this work in order to develop an efficient and robust mobile AR application.

2) Feature Description: Feature descriptors of the keypoints need to be constructed so that it can detect and match the features across the images. The description of each features must be unique and also persistent under all viewpoints. The idea that identified keypoints are invariant suggests that the small patches around them also have some characteristics that are unchangeable to rotation, brightness and scale. Hence, the mission of the descriptors is to capture these properties. Feature descriptors are grouped by two categories; floating-point descriptors and binary descriptor.

One of the most well-known floating-point keypoint descriptors is SIFT. This descriptor detects keypoints using Difference of Gaussians (DoG) [15]. Although SIFT was released in 2004, it still produced results that compete with state-of-the-art techniques. Apart from SIFT itself, several modified SIFT-like descriptors have been released, such as PCA-SIFT [20]. SURF almost matched the quality of SIFT but the use of integral images accelerates the gradient computations [16]. To date, the SURF descriptor is considered to be the most well-known alternative to SIFT. Both SIFT and SURF have well performance with their high distinctiveness and robustness in various of computer vision or image recognition applications [21], [22]. However, the floating-point descriptors still required high computation time for real-time applications, particularly for those application running on limited memory capacity and computing power such as tablet and smart phone [23]. Therefore, binary descriptors are designed to fill this gap.

Binary descriptors that achieved a compact storage and quick runtime are become increasingly popular due to fast evolvement of real-time application [24]. They show the similar quality as SIFT-like descriptors but at significantly lower computational costs and required a small amount of memory. Hamming distance is used to match binary descriptor due to each bit in the binary descriptor is independent. The four most promising binary feature descriptors are BRIEF [24], ORB [25], BRISK [26] and FREAK [27]. Hamming distance can be calculated effectively because the distance between descriptors were calculated by using XOR operation. Binary strings were generated by comparing the intensity of each pixel in the image. Binary string representing the area around the keypoint will be encoded in a string of "0" or "1". Generally, single bit of a binary descriptor are calculated by comparing the intensity value of point x in a sampling pair with the intensity value of point y in the pair. A single bit of a binary descriptor B on patch p can be calculated using Equation (1).

$$B(p;x,y) := \begin{cases} 1 : I(p,x) < I(p,y) \\ 0 : \quad \text{otherwise} \end{cases} \quad (1)$$

where $I(p,x)$ is the pixel intensity at point x of a sampling pair and $I(p,y)$ is the pixel intensity at point y of the sampling pair. A binary feature descriptor can be formed by concatenating the bits formed by B, as shown in Equation (2).

$$\sum_{1 \le i \le n} 2^{i-1} B(p;x,y), \quad (2)$$

where the n value for BRIEF and ORB is 256, while for BRISK and FREAK, it is 512. Based on previous researchers, FREAK descriptor has been identified as the most suitable descriptor for AR application [28]. FREAK descriptor was able to perform in shortest computation time and robust to rotation, scale and illumination invariance. Hence, this work is implement FREAK descriptor to describe image feature.

3) Feature Matching: Both descriptors obtained from reference image and query image are matched by comparing the distance between the descriptors using some metric [29].Euclidean distance is used to match features produced by floating-point descriptor (SIFT and SURF) as equation below [30]:

$$Euclidean\ Distance = \sqrt{\sum_{n=1}^{64} (Reference\ Descriptor_n - Query\ Descriptor_n)^2} \quad (3)$$

The smaller the Euclidean distance between reference descriptor and query descriptor, the more similar feature they have. Given two sets of descriptors, the best match in the first set (query image) is the descriptor that yields the lowest distance in the second set (reference image). Matching every descriptor in the reference image to every descriptor in query image will return the most matches.

Hamming distance is usually used to match features obtained from binary descriptor (BRIEF, ORB, BRISK and FREAK). Hamming distance is defined as the number of bits that differ between the strings. The distance can be computed quickly by performing an XOR (exclusive-or) operation on the two strings and counting the number of bits as Equation (4).

$$Hamming\ Distance = POPCNT\ (Reference\ Descriptor\ \hat{}\ Query\ Descriptor) \quad (4)$$

Logical operation such as XOR are very fast and most processor architectures have a built-in instruction for counting the number of "1" in a binary string (known as population count instruction). This makes matching binary descriptors can perform in shorter time compared to matching floating-point descriptors. The first step to obtain Hamming distance is to compare the first two bits in each string. If both bits are the same, a "0" will be recorded for that bit. If they are different, a "1" will be recorded. Then compare each bit in succession and record either "1" or "0" as appropriate. For example;

String 1: 1100 0110 0011;
String 2: 1000 0111 1011

and the results recorded are 0100 0001 1000. The last step in Hamming distance is to do addition mathematical operation for the recorded results which is $0+1+0+0+0+0+0+1+1+0+0+0=3$. As a result, the Hamming distance for these two string is 3. Similar to Euclidean distance, the smaller the Hamming distance, the more similar the feature. Hamming distance is used in this work since binary descriptor is used to describe feature.

B. Mobile Augmented Reality

A mobile AR application was developed using Eclipse software in HTC One X+ android smart phone. Most current available AR technologies support iOS and Android devices; however there is currently no standard library for use in mobile AR applications. OpenCV is rapidly becoming the accepted standard for computer vision application development and it can be used rapidly expandable AR frameworks. OpenCV is a library used for image processing. Hence, the image recognition algorithms used in this work are obtained from OpenCV 2.4.9 library. A simple flow to develop mobile AR application is shown in Fig. 1.

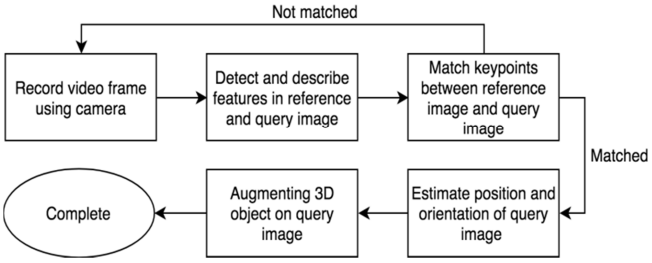

Fig. 1 Mobile AR application flow

The first step in mobile AR application is to record video frame using mobile's camera. The features in the reference image and query image are detected and described using feature detector and feature descriptor respectively. After successfully obtain keypoints from both image, the keypoints will be matched using a matcher. If the matching percentage between feature from reference image and query image is less than a threshold, the process need to repeat step one which is recording video frame. The process of estimating position and orientation of query image will take place if the matching percentage between feature from reference image and query image is more than a threshold. The last step in mobile AR application is to augment a 3D virtual object on top of query image. The mobile AR application development process can be described in more detail by dividing it into two parts, one is offline image recognition and the other is real time image recognition. Fig. 2 shows the proposed mobile AR application process.

1) Offline Image Recognition Process: Offline image recognition happens before the mobile phone camera work and does not happen in real time. All the image recognition algorithm are provided in OpenCV 2.4.9. OpenCV 2.4.9 library are imported into the programme. The reference image needs to be converted to gray scale image before performing the image recognition function because AGAST

and FREAK algorithm can only recognized gray scale images. Therefore, the function “cvtColor()” with parameter COLOR_RGBA2GRAY needs to be implemented to the reference image and stored in “referenceImageGray”.

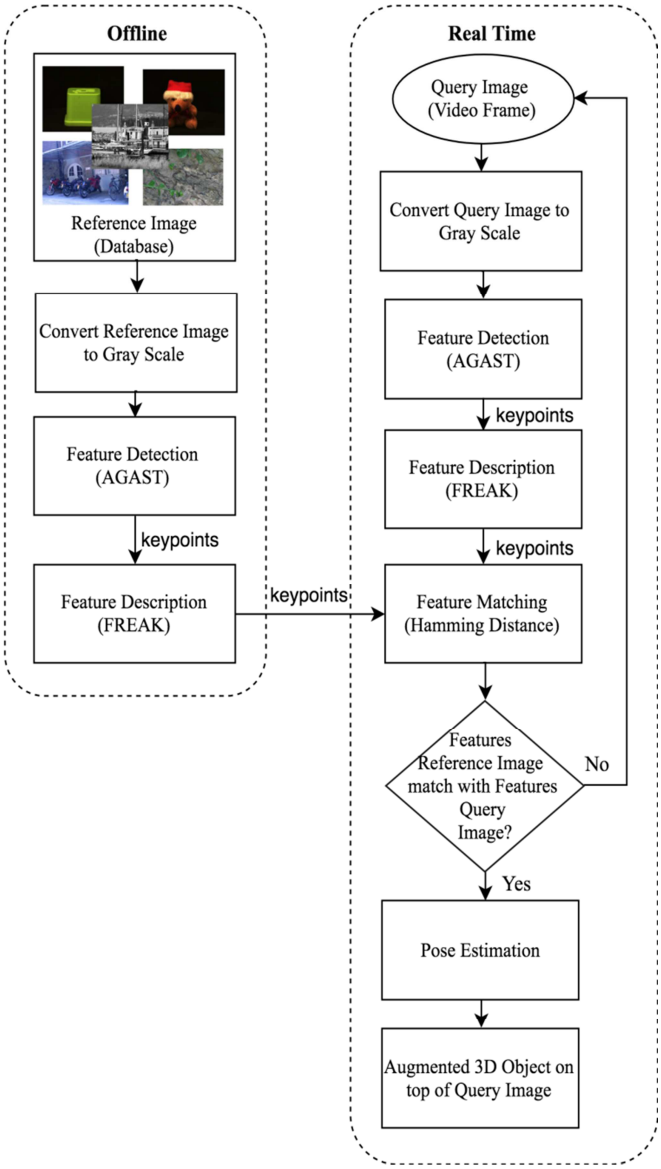

Fig. 2 Mobile AR Application Development Process

Offline image recognition is performed to detect and describe the features of images that have been stored in the database (“referenceImageGray”). Each reference image will be detected and described before the matching process between the reference image and query image. “FeatureDetector.create()” and “FeatureExtractor.create()” function are implemented in the programme and declare AGAST detectors and FREAK descriptor as the detector and descriptor algorithm. These declared algorithms are used in both parts (offline and real time) as offline and real time parts are using the same detector and descriptor. After declaring the algorithm, the next step is the process of detecting and describing the features of the reference image using the "detect ()" and "compute ()" functions. The detected reference image feature is stored in "mReferenceKeypoints". The descriptor describes the features stored in "mReferenceKeypoints" and store the features described in "mReferenceDescriptors". Fig. 3 shows the pseudocode used to detect and describe features in database (reference image).

```
Programme detect and describe features in database:
Declare detector as AGAST;
Declare descriptor as FREAK;
Convert RGB reference image to gray scale reference image;
Detect features in gray scale reference image using detector;
Describe features detected from gray scale reference image using descriptor
```

Fig. 3 Pseudocode used to detect and describe feature in database (reference image)

2) Real Time Image Recognition Process: Real time image recognition happens when mobile phone camera is working it has some processes which are similar to offline image recognition. Real time image recognition also includes feature detection and feature description process. However real time image recognition takes place in real time or when mobile phone camera is working. The first step is to make sure real time image recognition is recording a video frame with the mobile phone camera. This process can be taken by implementing "CameraBridgeViewBase" function. Camera configurations can be set for high quality video 640 × 480, medium quality video 480 × 360 or low quality video 192 × 144 using the "setMaxFrameSize ()" function provided by the OpenCV library. Computation time is directly affected by reducing or increasing the video’s quality. The higher the video’s quality, the longer the computation time. The video quality used in this work are high quality video which is 640 x 480. The next process is the same as offline image recognition which is to convert the images obtained through video frame recording (query image) to gray scale query image and stored in "inputImageGray". The next step after getting a gray scale query image is to detect and describe the image features. This process is the same as the offline part but this is to detect and describe the feature from query image. Functions used to detect and describe features are "detect ()" and "compute ()". The detected features obtained from query image are stored in "mInputKeypoints" and the described features are stored in "mInputDescriptors". The following step is to match between the two descriptor sets (feature descriptors obtained from the query image and the reference image). "BRUTEFORCE_HAMMING" , where hamming distance is declared as the matching algorithm. "DescriptorMatcher.create ()" function needs to be implemented in order to carry out the matching process. The matching algorithm used in this work is hamming distance because the detector and descriptor algorithm used are binary. The matching process between reference image descriptors and query image descriptors is done by implementing the "match ()" function and stored in "mMatches". Fig. 4 shows the pseudocode used to match descriptor from reference image and query image.The total number of matches are always lower than the number of feature descriptors since not all query image descriptor will have best match with the reference image descriptors. Pose

estimation process is carried out where the features in query image are matched with reference image in database. If the number of matching is lower than the threshold, the process will be repeated where the video frame is taken with the camera. Pose estimation is performed to determine the position of a virtual object on top of the input image.

```
Programme matching descriptor from reference image and query image:
Detect features in gray scale query image using detector;
Describe features detected from gray scale query image using descriptor
Declare descriptor as HAMMING MATCHER;
Match features describe from reference image and query image;
```

Fig. 4 Pseudocode used to match descriptor from reference image and query image

The next step is to implement "myPoseEstimation ()" function to determine the 3D object positions on top of query images. In addition, the "CameraProjectionAdapter" adjustment tool is used to provide "Camera.Parameters" object and get one projected matrix in OpenCV or OpenGL format. All the data needed to build projection matrix is stored in "CameraProjection". The next step is to find the position and rotation of the query image based on the projection matrix. The "Calib3d.solvePnP ()" function is used to perform the positioning and rotation of the query image. This function stored position and rotation results in two separate vectors. The position matrix and the rotation results are manually diverted to 16 bits of the appropriate layout for presenting 3D objects. After this process is successful, a 3D cube will be augmented on top of query image (Fig. 5).

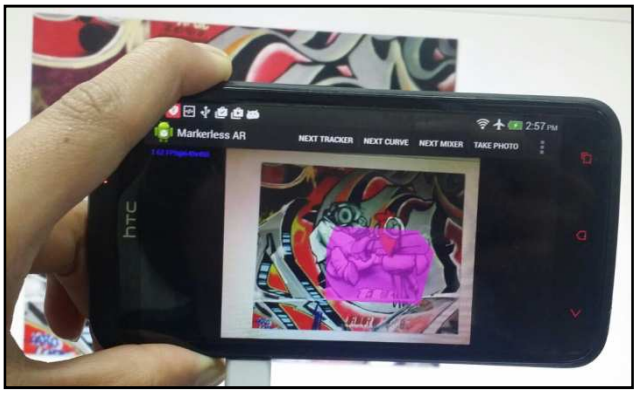

Fig. 5 3D virtual cube is successfully augmented on top of query image

C. Dataset

The evaluation was implemented on HTC One X+ android smart phone. It has a built-in camera and is able to record video with 1080 pixels at 28 fps or 720 pixels at 30 fps which fulfils the basic requirements for successful implementation of mobile AR application. The image recognition algorithms are obtained from OpenCV 2.4.9 library. The dataset used for evaluation in this research is the well-known dataset introduced by researcher [13] which had been used by most researchers [31]–[33] shown in Fig. 6.

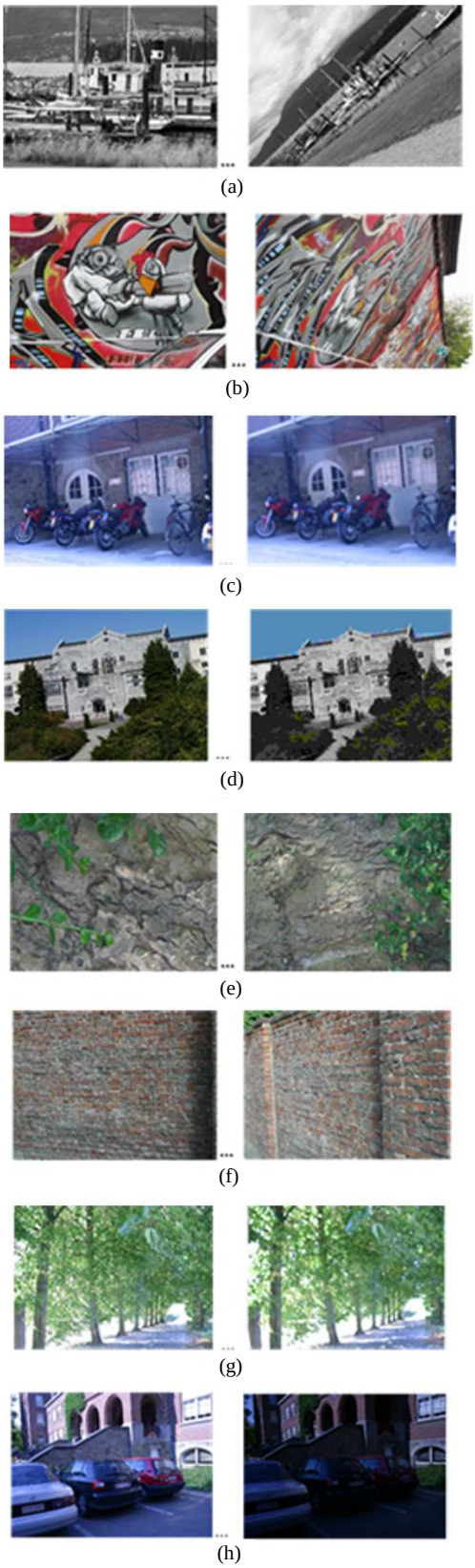

Fig. 6 Mikolajczyk Dataset

This dataset consists of image sets with different photometric and geometric and transformations (rotation, scale, viewpoint, blur and illumination) and with different scene types (textured and structured) as shown in Table I. In the case of rotation, scale change, blur and view point

change, two different scene types are used; textured and structured. For illumination case, the light changes are proposed by varying the camera aperture. Each image set contains six images with a photometric distortion or gradual geometric.

TABLE I
TRANSFORMATION AND SCENE TYPE FOR EACH IMAGE

No	Image	Scene Type	Transformation
a.	Boat	Structured	Scale, Rotation
b.	Graffiti	Structured	View Point
c.	Bikes	Structured	Blur
d.	UBC	Structured	JPEG Compression
e.	Bark	Textured	Scale, Rotation
f.	Wall	Textured	View Point
g.	Trees	Textured	Blur
h.	Leuven	Structured	Illumination

D. Efficiency

Efficiency is generally defined as the ability to recognize corresponding points between consecutive frames (reference image and query image) in the shortest time possible. It is often interchanged with words such as "fast" and "speed".

The most important evaluation for mobile AR application is its computation time for the whole process. Computation time of an efficient mobile AR application should function in short time to ensure the application is able to run in real time. The total computation time to perform augmentation process is calculated by adding the computation time of each individual process such as start video frame, feature detection, feature description, feature matching, pose estimation and 3D object augmentation. The computation time for each individual process is calculated using Equation (5). Let a process start time denote as T_s, a process end time denote as T_e and total process time denote as T_t. If $f(x)$ is the function for each process (start video frame, feature detection, feature description, feature matching, pose estimation and 3D object augmentation), then computation time for function $f(x)$ is as below:

$$f(x) = T_t(T_e - T_s) \quad (5)$$

E. Robustness

Robustness can be defined as accurate recognition of corresponding points between two frames (reference image and query image) in large changes in scale, rotation and illumination [34]. "Accuracy" is another term often used to describe robustness of the image recognition techniques. The robustness or accuracy rate of a mobile AR application is calculated using the equation below:

$$Accuracy\ Percentage = \frac{Number\ of\ Correct\ Matches}{Number\ of\ Matches} x100\% \quad (6)$$

The section discussed the experimental results of the evaluation. The performance of mobile AR application is measured in terms of efficiency (computation time) and robustness (rotation, scale and illumination invariance).

III. RESULTS AND DISCUSSION

A. Computation Time

Computation time to carry out detection, description and matching process is recorded every 500 keypoints and repeated 50 times. The results shown in Table II are the average computation time used to perform each process and total computation time of the entire mobile AR application process. Computation time for each process is calculated using Equation (5).

TABLE II
TRANSFORMATION AND SCENE TYPE FOR EACH IMAGE

Process	Time (ms)
Capture from video frame	1.3
Convert image to gray scale	2.3
Detect 500 keypoints	13.8
Describe 500 features	4.3
Matching 500 keypoints	1.9
Pose Estimation	4.1
Augmented 3D object	1.4
Total Computation Time	**29.1**

The mobile AR application is able to perform in short computation time which is 29.1ms. All image recognition algorithms used in this work are able to function in real time and as a result the mobile AR application is also able to work in real time where the computation time is less than 100ms.

B. Accuracy

Accuracy of mobile AR application in terms of scale, rotation and illumination are evaluated using Mikolajczyk dataset mentioned in Section IV. The accuracy of the mobile AR application is calculated using Equation (6) and repeated 50 times for each evaluation. The accuracy of each invariance is concluded in Table III.

TABLE III
ACCURACY OF EACH INVARIANCES

Invariance	Accuracy
Brightness	83.87%
Scale	89.76%
Rotation	87.71%

The mobile AR application is able to obtain good accuracy in terms of brightness, scale and rotation invariance. All the three invariances are able to perform higher than 80%.

IV. CONCLUSION

This work presents an efficient and robust mobile AR application. The image recognition used in detection, description and matching are AGAST, FREAK and Hamming distance respectively. From the evaluation result, the mobile AR application is able to work in real time and is robust to scale, brightness and rotation invariance. The mobile AR application performs in a short time and have a

good recognition accuracy percentage up to 83.67% even though the image had experience changes in brightness, scale and rotation. Therefore, the development process and algorithm suggested in this work guarantees a good performance of mobile AR application.

ACKNOWLEDGMENT

This research work is supported by UKM Research Grant (GGPM-2018-011).

REFERENCES

[1] R. T. A. Azuma, "Survey of Augmented Reality," Presence: Teleoperators and Virtual Environments, vol. 6, no. 4. pp. 355–385, 1997.

[2] D. Nincarean, M. B. Alia, N. D. A. Halim, and M. H. A. Rahman, "Mobile Augmented Reality: The Potential for Education," Procedia - Soc. Behav. Sci., vol. 103, pp. 657–664, 2013.

[3] M. Pu, N. A. A. Majid, and B. Idrus, "Framework based on Mobile Augmented Reality for Translating Food Menu in Thai Language to Malay Language," Int. J. Adv. Sci. Eng. Inf. Technol., vol. 7, no. 1, pp. 153–159, 2017.

[4] M. J. Sadik and M. C. Lam, "Stereoscopic Vision Mobile Augmented Reality System Architecture in Assembly Tasks," J. Eng. Appl. Sci., vol. 12, no. 8, pp. 2098–2105, 2017.

[5] H. Arshad, M. C. Lam, W. K. Obeidy, and S. Y. Tan, "An Efficient Cloud based Image Target Recognition SDK for Mobile Applications," Int. J. Adv. Sci. Eng. Inf. Technol., vol. 7, no. 2, pp. 496–502, 2017.

[6] N. C. Hashim, N. A. A. Majid, H. Arshad, and W. K. Obeidy, "User Satisfaction for an Augmented Reality Application to Support Productive Vocabulary Using Speech Recognition," Adv. Multimed., 2018.

[7] L. W. Shang, M. H. Zakaria, and I. Ahmad, "Mobile phone augmented reality postcard," J. Telecommun. Electron. Comput. Eng., vol. 8, no. 2, pp. 135–139, 2016.

[8] H. Arshad, S. A. Chowdhury, L. M. Chun, B. Parhizkar, and W. K. Obeidy, "A freeze-object interaction technique for handheld augmented reality systems," Multimed. Tools Appl., 2016.

[9] D. Wagner and D. Schmalstieg, "History and Future of Tracking for Mobile Phone Augmented Reality," in 2009 International Symposium on Ubiquitous Virtual Reality, 2009, pp. 7–10.

[10] W. K. Obeidy, H. Arshad, S. Y. Tan, and H. Rahman, "Developmental Analysis of a Markerless Hybrid Tracking Technique for Mobile Augmented Reality Systems," in Advances in Visual Informatics, 4th International Visual Informatics Conference, IVIC 2015, 2015, pp. 99–110.

[11] H. Uchiyama and E. Marchand, "Object Detection and Pose Tracking for Augmented Reality: Recent Approaches," Found. Comput. Vis., pp. 1–8, 2012.

[12] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in Procedings of the Alvey Vision Conference 1988, 1988, p. 23.1-23.6.

[13] C. Mikolajczyk, K.Schmid, "A performance evaluation of local descriptors," IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 10, pp. 1615–1630, 2005.

[14] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in Proceedings of the IEEE International Conference on Computer Vision, 2005, vol. II, pp. 1508–1515.

[15] G. Lowe, "SIFT - The Scale Invariant Feature Transform," Int. J., vol. 2, pp. 91–110, 2004.

[16] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," Comput. Vis. Image Underst., vol. 110, no. 3, pp. 346–359, 2008.

[17] J. R. Quinlan, "Induction of Decision Trees," Mach. Learn., vol. 1, no. 1, pp. 81–106, 1986.

[18] H. Zhang, J. Wohlfeil, and D. Grießbach, "Extension and evaluation of the agast feature detector," ISPRS, vol. III, no. 4, pp. 133–137, 2016.

[19] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2010, vol. 6312 LNCS, no. PART 2, pp. 183–196.

[20] Y. K. Y. Ke and R. Sukthankar, "PCA-SIFT: a more distinctive representation for local image descriptors," Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, 2004. CVPR 2004., vol. 2, pp. 2–9, 2004.

[21] K. Mikolajczyk, A. Zisserman, and C. Schmid, "Shape recognition with edge-based features," in Procedings of the British Machine Vision Conference 2003, 2003, p. 79.1-79.10.

[22] T. Quack, H. Bay, and L. Van Gool, "Object Recognition for the Internet of Things," First Int. Conf. Internet Things (IoT 2008), vol. 4952, pp. 230–246, 2008.

[23] L. Naimark and E. Foxlin, "Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker," in Proceedings - International Symposium on Mixed and Augmented Reality, ISMAR 2002, 2002, pp. 27–36.

[24] 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, pp. 778–792.

[25] 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, pp. 2564–2571.

[26] 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, pp. 2548–2555.

[27] 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, pp. 510–517.

[28] S. Y. Tan, H. Arshad, and A. Azizi, "Evaluation on Binary Descriptor in Markerless Augmented Reality," in The 3rd National Doctoral Seminar on Artificial Intelligence Technology, 2014, pp. 1–6.

[29] A. Ufkes and M. Fiala, "A markerless augmented reality system for mobile devices," in Proceedings - 2013 International Conference on Computer and Robot Vision, CRV 2013, 2013, pp. 226–233.

[30] P. E. Danielsson, "Euclidean distance mapping," Comput. Graph. Image Process., vol. 14, no. 3, pp. 227–248, 1980.

[31] T. Tian, F. Yang, K. Zheng, and Q. Gao, "A Fast Local Image Descriptor Based on Patch Quantization," in International Conference on Human Centered Computing, 2017, pp. 64–75.

[32] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017, vol. 2017–Janua, pp. 3852–3861.

[33] D. J. Matuszewski, A. Hast, C. Wahlby, and I.-M. Sintorn, "A short feature vector for image matching: The Log-Polar Magnitude feature descriptor," PLoS One, vol. 12, no. 11, 2017.

[34] W. K. Obeidy, "A Markerless Hybrid Tracking Technique To Improve The Efficiency And Robustness Of Mobile Augmented Reality," Universiti Kebangsaan Malaysia, 2014.