symmetry

MDPI

Article

Optional Frame Selection Algorithm for Adaptive Symmetric Service of Augmented Reality Big Data on Smart Devices

HwiRim Byun [1], Jong Hyuk Park [2] and Young-Sik Jeong [1,*]

[1] Department of Multimedia Engineering, Dongguk University, Seoul 04620, Korea; hazzzly@dongguk.edu
[2] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea; jhpark1@seoultech.ac.kr
* Correspondence: ysjeong@dongguk.edu; Tel.: +82-2-2260-3374

Academic Editor: Doo-Soon Park
Received: 11 April 2016; Accepted: 17 May 2016; Published: 23 May 2016

Abstract: Following recent technological advances in diverse mobile devices, including smartphones, tablets and smartwatches, in-depth studies aimed at improving the quality of augmented reality (AR) are currently ongoing. Smartphones feature the essential elements of AR implementation, such as a camera, a processor and a display in a single device. As a result, additional hardware expansion for AR implementation has become unnecessary, popularizing AR technology at the user level. In the early stages, low-level AR technology was used mainly in limited fields, including simple road guides and marker-based recognition. Due to advances in AR technology, the range of usage has expanded as diverse technologies and purposes are combined. Users' expectations of AR technology have also increased with this trend, and a high quality of service (QoS), with high-resolution, high-quality images, is now available. However, there are limitations in terms of processing speed and graphic treatment with smart devices, which, due to their small size, have inferior performance compared to the desktop environment when processing data for the implementation of high-resolution, high-quality images. This paper proposes an optional frame-selection algorithm (OFSA), which eliminates the unnecessary work involved with redundant frames during rendering for adaptive symmetric service of augmented reality big data on smart devices. Moreover, the memory read-write delay of the internally-operating OFSA, is minimized by adding an adaptive operation function. It is possible to provide adaptive common AR images at an improved frame rate in heterogeneous smart devices with different levels of performance.

Keywords: augmented reality; heterogeneous adaptiveness; smart device; frame rate; snapshot

1. Introduction

Augmented reality (AR) provides additional information by overlaying virtual objects, created by a computer, onto real objects. Tom Caudell coined the term AR in 1990, and its meaning was defined by Milgram, Takemura, Utsumi and Kishino in 1994 [1–3]. Diverse research and development of AR began after Klopfer expanded it in 2008 to combine information with the real world [4,5]. Recently, studies have been done on AR that are aimed at achieving the optimal performance of diverse smart devices, including mobile phones, tablets iPods, iPads and iPhones [6–12]. Due to the continuous growth of the application field of AR, continuous research and development aimed at the provision of high-resolution, high-quality images are in progress [13–15]. However, AR images with high resolution and high quality cause delays and sporadic frame drops in smart devices that have generally used specifications, as they induce heavy loads at the processor level. Although there have been many studies aimed at achieving a higher frame rate by creating an efficient mechanism, problems arise due to the need for additional sensors and an Internet connection. This paper proposes

an optional frame-selection algorithm (OFSA) to achieve a better frame rate in low-specification devices for adaptive symmetric service of augmented reality big data on smart devices. OFSA involves the differences between changing AR objects due to minute camera movements due to limited display resolution or the human body's recognition and acceptable errors [16,17]. Therefore, users cannot distinguish a change in camera pose below a certain level. Hence, in cases of small frames with a change of camera pose smaller than the threshold, the processing of each frame can be minimized by maintaining the AR object of the previous frame without executing additional rendering. Figure 1 shows a frame processed by OFSA for adaptive symmetric service of augmented reality big data on smart devices that are actually operational. The frames were processed without omissions on laptop and desktop machines, which have better performance than the requirement. However, some frames were omitted on tablet PCs and smartphones, falling short of the recommendation. The omitted frames are those that differ from the previous frame at a level below the threshold; they are regarded as useless frames that do not require rendering. In this way, OFSA is capable of common AR implementation in smart devices that have diverse functions.

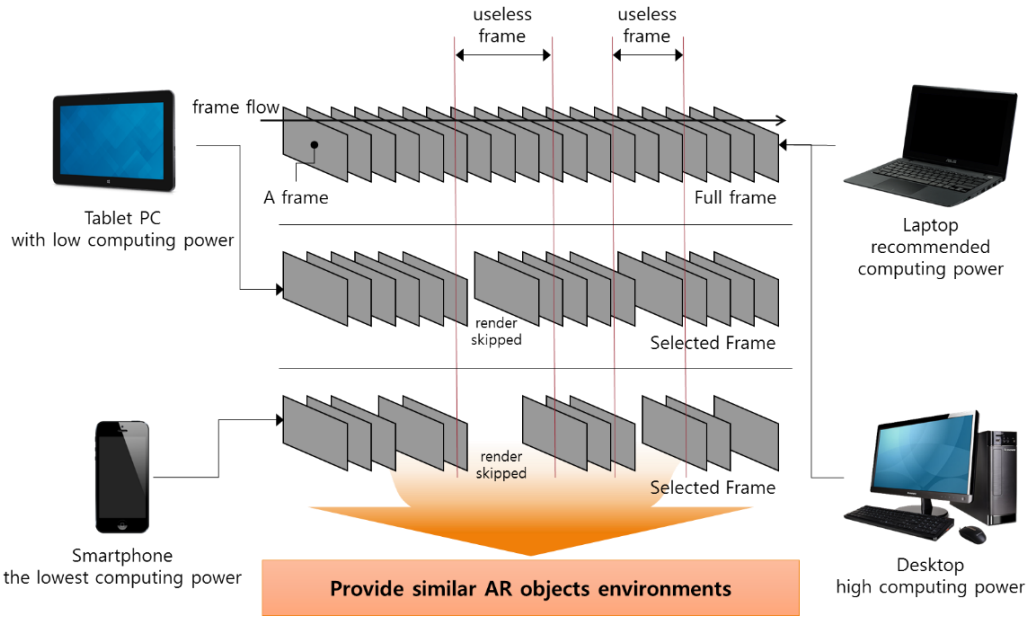

Figure 1. Basic frame processed by the optional frame-selection algorithm (OFSA) for adaptive symmetric service. AR, augmented reality.

This paper is structured as follows. Section 2 introduces related literature regarding the achievement of higher frame rates in AR. Section 3 explains the OFSA scheme, similarity calculation, unit correction and adaptive operation. Section 4 describes the design of monitoring OFSA (mo_OFSA) for the applications of monitoring and reviewing. Section 5 implements mo_OFSA, and Section 6 proves the performance improvement by conducting a performance evaluation. Finally, the paper concludes with an overall summary and future research topics.

2. Related Works

This section introduces studies that are focused on securing higher frame rates to enhance the sense of immersion and reality of AR. Previous studies mainly tracked pre-processing, post-processing or distributed processing. Data correction with measurements using sensors, the image process that recognizes an object and predicts its movement, and distributed processing that uses multiple threads or the cloud environment are used as schemes for achieving high frame rates applicable to diverse smart devices. These are accompanied by a number of constraint conditions, such as the necessity of an Internet connection or additional sensors. Table 1 summarizes the existing literature.

Table 1. Comparison with exiting research.

Type of Research	Functions and Operation Mechanisms
Motion State Estimation (MSE) [6]	• Proposes a scheme that addresses the problem of delays in tracking due to motion blur that occur because of the movement of the AR marker. • Capable of securing more frames than the SIFT and SURF algorithms during the same amount of time in an environment of moving markers by achieving faster tracking speed through a prediction of marker movement divided into stable, slow, linear and non-linear. • Applicable only in devices in which markers move in identical patterns.
Augmented reality with high frame rate for low computational power devices [7]	• Proposes a scheme for obtaining a fast camera pose of AR in devices with low computing power. • Parallel processing of three major works of AR (acquisition, tracking and rendering) by using a thread. • Proven effective performance from the aspect of frame rate and temporal jitter. • Requires an environment where multi-thread implementation is possible.
Model-based tracking [8]	• Proposes a 3D model-based estimation algorithm for the purpose of securing more frames by diminishing the time in the tracking stage of markerless AR. • Estimates a complete 3D model by predicting the outline information of the tracked object. • Requires an additional process for 3D model estimation.
Real-time hybrid-based method for tracking unknown environments in markerless augmented reality [9]	• Proposes a scheme that enhances the speed of synchronization process computation of the image received to the rendering of the AR model and camera. • Implements reinforcement in computer vision using the acceleration and gyroscope sensors. • Obtains results that are as much as five-times faster than the process computation of computer vision-based tracking. • Applicable only in devices that have accelerometer and gyroscope sensors.
Histogram of Intensity PatcheS (HIPS) [10]	• Proposes a scheme that uses the distributed processing of the Kinect for the purpose of constituting faster AR using the limited computing performance of smart devices. • Capable of collaboration between more than two devices. • Internet environment and Kinect for rapid image processing are essential.
Mobile augmented reality based on cloud computing [11]	• Achieves a high frame rate by moving the process of matching the tracked object to the AR object in a markerless AR environment into a cloud environment. • The Internet environment for the cloud is essential.

This paper explains the problem of requiring an Internet environment and additional sensors. As OFSA has better performance at the algorithm level, it can be applied to all smart devices without requiring other sensors or environments.

3. Scheme of OFSA

OFSA, as proposed in this paper, compares the current frame to the previous frame and omits rendering when the movement of the AR object is below the threshold. As a result, the level of processing decreases, improving the frame rate. As more frequent image acquisition and tracking becomes possible, more seamless AR can be provided to users. OFSA computes a camera pose matrix based on a target object recognized in the frame and categorizes frames that require rendering and

frames that do not. By comparing the current frame and the key frame, it categorizes frames with significant movement as requiring rendering and those without significant movement as useless and not requiring rendering. The criteria of significant movement are determined according to a user-selected threshold. The fact that a camera pose matrix that constantly changes according to the movement of the camera does not result in a significant difference between the current frame and the key frame indicates that the change in the rotation angle and the location of the AR object is small.

Figure 2 shows non-OFSA and OFSA processing in an AR implementation through smart devices. Non-OFSA refers to an AR implementation scheme without OFSA application, following the process flow of the Vuforia API [18]. The process flow of AR implementation in the non-OFSA environment can be divided into three stages: image acquisition, in which the image is received from the input device, such as the camera; tracking, which derives the target object from the received image and computes the camera pose matrix; and rendering, which creates the AR object in an appropriate form on the current frame based on the camera pose matrix. In this process, OFSA comes after tracking and uses the camera pose matrix computed in the tracking process. OFSA not only determines whether or not rendering should be performed, but also stores a snapshot of the AR object after the rendering. Hence, the rendering process is included in OFSA. In contrast to non-OFSA, OFSA performs an additional process for each frame. However, it stores the OFSA value of the key frame and compares the OFSA value in each frame to reduce the load on the processor based on this computation. As such, the processor overload rate decreases by omitting the rendering of meaningless frames, achieving a higher frame rate. As the omitted frames have very small changes in camera pose, which users cannot visibly detect, an identical AR service can be provided without being restricted by the computing power.

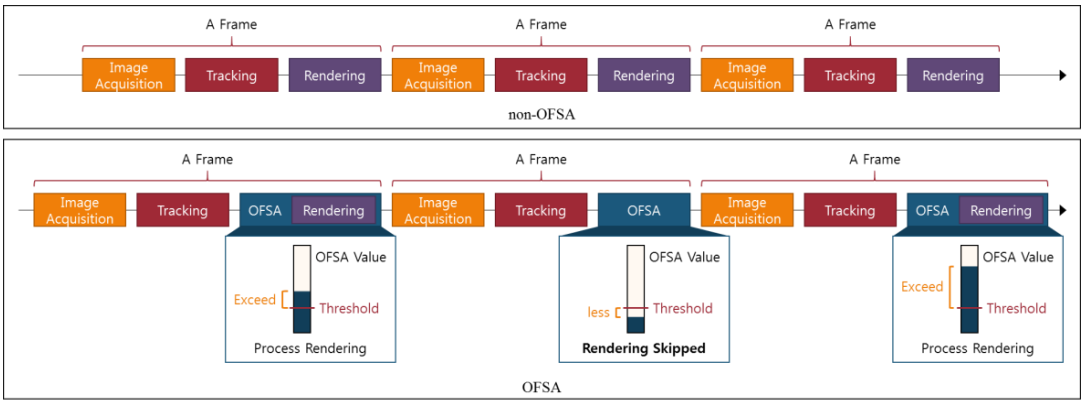

Figure 2. Processing of non-OFSA and OFSA.

The OFSA process consists of five steps (Figure 3):

Step 1: The current frame of the image is received from the smart device to obtain the camera pose matrix.

Step 2: Whether there is a key frame that was previously announced is determined. In the first entry to OFSA, there is no key frame, and hence, the current frame is registered as a key frame.

Step 3: The OFSA values of the current frame and the key frame are computed. This value numerically interprets whether there was a significant difference of movement between two frames.

Step 4: Rendering is determined by comparing the OFSA value and the selected threshold. If the OFSA value exceeds the threshold, the process for adaptive operation takes place. If it is smaller, it is regarded as a useless frame that does not have a large difference from the key frame, and hence, the previous snapshot is maintained.

Step 5: Whether the previous frame is a key frame is checked, to decrease delays due to memory read-writing by preventing frequent registration of key frames. If the previous frame is a key

frame, the non-OFSA process is engaged to remove the snapshot, and each frame rendering begins. Otherwise, the current frame is registered as a key frame.

```
procedure OFSA( CF : Current Frame, N : number of frame)
   CF = Current Frame Data from camera
   KF = Key Frame Data from calculate
   repeat
   /* calculate CF's camera pose matrix */
   CFcpm = Camera Pose Matrix(CF)
   if is KF existed then
      /* OFSAvalue calculate between CF and KF */
      get OFSAvalue( CFcpm, KFcpm)
      if OFSAvalue > Treshold then
         /* check previous frame was key frame */
         if nKF = N-1 then
            set KF as CF
            KFcpm = CPM(KF)
            nKF = N
         else
            goto register
         end if
      end if
   else
register :
      set KF as CF
      KFcpm = CPM(KF)
      nKF = N
      rendering AR object
      /* get present screen snapshot with AR object */
      take snapshot
   end if
   display snapshot
   until end of video
end procedure
```

Figure 3. Pseudocode of OFSA.

The OFSA value of the subsequent frames is calculated using a single key frame as a standard. The frame becomes a useless frame, and rendering is omitted when the value does not exceed the threshold. Useless frames can be determined without restrictions on their number between key frames. It is also possible to continuously determine only key frames, without useless frames. When a frame satisfies the threshold, it becomes a new key frame, as a standard. The interval between repeats continues until the received image finishes (Figure 3).

3.1. Similarity Calculation

The OFSA value numerically captures the range of movement between the key frame and the current frame by using the camera pose matrix [19,20]. This is a parameter that explains the conversion relationship between camera coordinates and world coordinates, which generally takes a three-row, four-column form in which the first, second and third columns have a rotation value and the fourth column has a translation value. OFSA searches for the target object in the tracking process every time a current frame is received, based on how the camera pose matrix is computed and stored. The camera pose matrix numerically captures the direction and angle of the camera shooting the image and consists of rotation values and translation values. To obtain the OFSA value, the camera pose matrix of the current frame and the key frame is divided into rotation and translation, and the difference of each element is computed to obtain the sum of the absolute value of each element. Here, as the rotation

and translation values have different units, the unit correction value is multiplied to minimize errors resulting from the unit difference.

$$Variation_{RT} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = Keyframe_{RT} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} - Currentframe_{RT} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (1)$$

Equation (1) shows the computation of the OFSA value. The calculated OFSA value contains numerical information about the range of movement between two frames. In the equation above, $Keyframe_{RT}$ and $Currentframe_{RT}$ refer to the camera pose matrix of the key frame and the current frame, respectively, and $Variation_{RT}$ refers to the range of change of the current frame using a key frame as a standard.

$$OFSA\ Value = \sum_{i=1}^{9} |mi| + \sum_{i=10}^{12} |mi| * Correction\ Value \quad (2)$$

Equation (2) converts the camera pose matrix into a single constant. Unit correction of rotation and translation values in the element of $Variation_{RT}$ is also conducted. *Correction Value* is multiplied by the sum of three elements of the matrix that indicate the translation, and the *OFSA value* is obtained by adding the sum of the rest of the zero elements that indicate the rotation value.

3.2. Unit Correction

As the rotation and translation values used in the computation of *OFSA values* have different units, a raw value cannot be used. Hence, an additional correction calculation is necessary for addressing the unit difference. The rotation value can have a value between 0° and 360° and is numerically expressed as a value between zero and one. The translation value has a direct relationship with resolution, with a range between zero and the maximum number of pixels of the frame length and width. Assuming the resolution of typical smart devices to be 800 by 400 pixels, the rotation value is negligible compared to the translation value. If the *OFSA value* is computed in a condition without unit correction, it can be sensitive to the change of translation, but it shows almost no response to the change of rotation. Hence, *Correction Value* is multiplied by the translation for correction to expand the numerical range of rotation, so that it can be the same as translation, or to set the range of translation value between zero and one.

$$Correction\ Value = \frac{1}{Frame\ Width\ +\ Frame\ Height} \quad (3)$$

Equation (3) is an equation for Correction Value, which makes the translation value have an identical scale to the rotation value through multiplication, where the frame width and height refer to the size of the currently-obtained image.

3.3. Adaptive Operation

In OFSA, the snapshot process performs the process of temporarily storing the rendered object in memory in pixel units, which causes a decreased frame rate by inducing a large amount of memory read-write. The number of call outs in the snapshot process increases, as pose changes that exceed the threshold occur more frequently. Hence, rendering is conducted in each frame by non-OFSA when there is a change that continuously exceeds the threshold. If a change smaller than the threshold occurs for more than two frames, OFSA is applied again by regarding the situation as stable.

4. Design of mo_OFSA

The Vuforia API was used in the design of an Android-based AR viewer application for the actual implementation and performance evaluation of mo_OFSA (monitoring_OFSA). mo_OFSA is designed to measure frames per second (FPS) by converting between non-OFSA and OFSA in a marker-based AR environment. An FPS level of 40 was maintained, since valid performance checks are difficult if the maximum FPS of 60 is exceeded in idle conditions in the testing environment of mo_OFSA

in AR models that overwrap the marker. Figure 4 shows the architecture of mo_OFSA. Image data received from the camera of the smart device is converted in frame units for computation by Vuforia's Frame Converter.

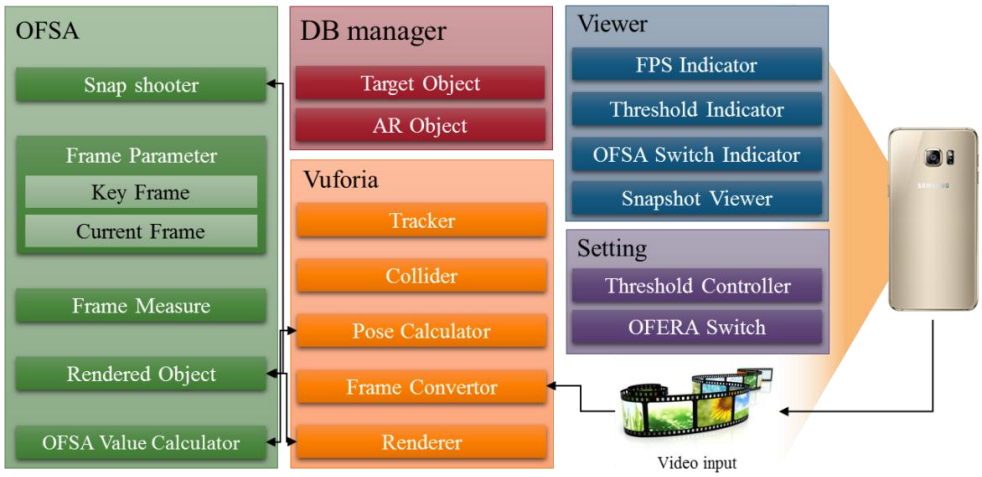

Figure 4. Architecture of monitoring OFSA (mo_OFSA).

Vuforia consists of Frame Converter; Tracker, which detects the target object from the converted image data; Collider, for multi-target AR; Pose Calculator, which computes the camera pose matrix of the frame using the recognized target object; and Renderer, for the rendering of AR objects. *OFSA* consists of Snap Shooter, which stores images in memory to display the rendered object in useless frames without an additional rendering process; Frame Parameter, which stores the parameters of each frame for comparisons between key frames and current frames; Frame Measure, which processes the comparison computation of OFSA values; Rendered Object, which stores the completely rendered object in case rendering is processed after the renewal of the key frame; and OFSA Value Calculator, which computes the OFSA value using the data from Frame Converter. *DB manager* stores the model and object required for AR implementation and includes the AR object, which stores the rendered AR object, and the target object, which stores information about a real object, onto which the relevant object will be overlaid. *Viewer* provides the user interface (UI). It consists of FPS Indicator, which directly compares the frame rate between OFSA and non-OFSA; Threshold Indicator, which checks the threshold value of the current user setting; OFSA Switch Indicator, which checks the setting of OFSA and non-OFSA; and Snapshot Viewer, which displays snapshots of the OFSA process. *Setting* determines and stores the detailed OFSA settings. The setting interface includes Threshold Controller, which determines the snapshot in the OFSA process according to the difference in camera poses between the current frame and the key frame, and OFSA Switch, which switches between the implementation of non-OFSA and OFSA.

5. Implement of mo_OFSA

Figure 5 exhibits the composition of mo_OFSA implemented for the performance measurements of OFSA. Its structure features four overlaid layers, which are described in ① of Figure 5. The first layer is the camera layer, where the image received from the camera is displayed. The second layer is the convectional layer, which displays the target object that was recognized by non-OFSA on the AR object. The third layer is the snapshot layer, which displays the snapshot stored by OFSA. The fourth layer is capable of performing switches between non-OFSA and OFSA in the adaptive OFSA environment without additional processes, as it consists of the user interface (UI) layer. Figure 5 ② shows a checkbox for switching between OFSA and non-OFSA, where AR is implemented by non-OFSA if unchecked, while the adaptive OFSA process is applied if checked. Figure 5 ③ shows an indicator for controlling the threshold of OFSA, where the sensitivity of OFSA is set. The initial value is 30, and mo_OFSA

supports a control range between zero and 100. Figure 5 ④ shows an FPS indicator, which presents the frame rate of mo_OFSA as a 0.5-s renewal and records it for AR service analysis. Figure 5 ⑤ shows the actual application screen for when the change of camera pose is smaller than the threshold in the OFSA environment. Although there is a minute change in the angle of the camera view, the AR object is maintained as the identical model through the snapshot, and the FPS also shows the same level as under the idle condition. Figure 5 ⑥ shows a case of the camera pose change being larger than the threshold in the OFSA environment. A new object that is different from the AR object in Figure 5 ⑤ is rendered to be registered in the snapshot and displayed.

Figure 5. Composition of mo_OFSA.

Figure 6 shows frame flow, which checks the frame selection of OFSA using mo_OFSA. An image was used in which the camera moved while centered on the target object. Each frame has a 1-s distance in the image, which shows how the AR model is processed in non-OFSA and OFSA environments. In non-OFSA, the AR model was rendered in each frame according to the frame flow. In OFSA, ①, ②, ③ and ④ in Figure 6 display identical AR models using a single snapshot. This is because the difference in the camera pose matrix among the four frames from ① is small. However, ⑤ and ⑥ in Figure 6 are displayed by storing the snapshot that differs from the previous one after detecting a change in the camera pose matrix that exceeds the threshold.

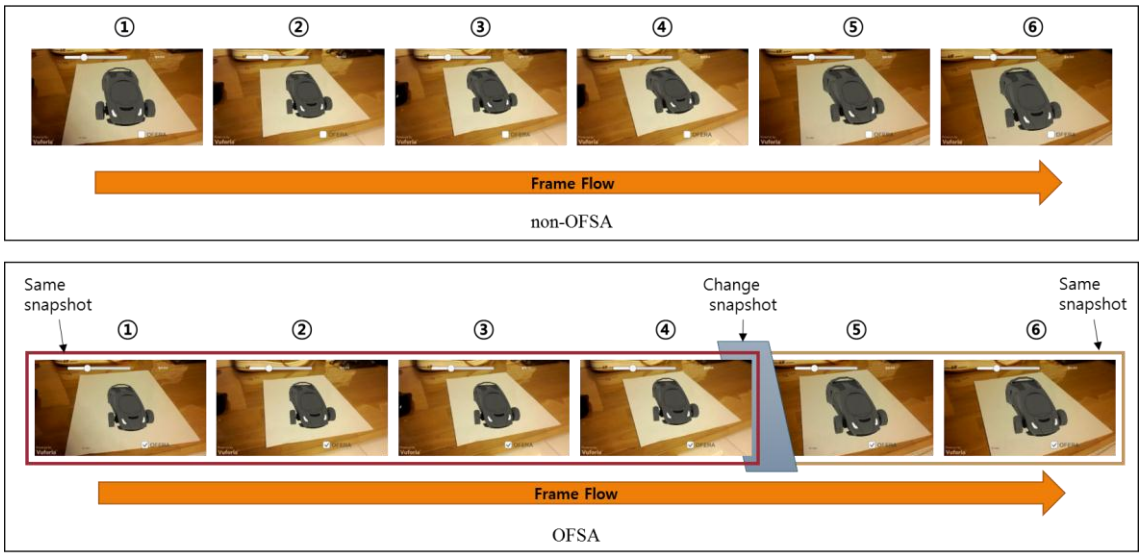

Figure 6. Comparison frame flow between non-OFSA and OFSA by mo_OFSA.

6. Performance Evaluation

For the performance evaluation of the image analysis of non-OFSA and OFSA, the additional frame rate that can be achieved was used as a criterion for performance analysis, and FPS was measured twice per second. The relative performance evaluation was conducted by calculating the average and median FPS based on the measured results. Using the moving condition of the target object as a standard, Stable refers to a situation in which the target object is stopped in the frame without moving; Slow Moving refers to the target object moving slowly; and Fast Moving refers to the target object moving quickly. The direction of the target object's movement is random, rather than linear or nonlinear. For the homogeneity of the image, a 5-s image length composed of identical environments was used for each standard. All images began with no target object in the frame, and the target object was recognized after 1 s. In Slow Moving and Fast Moving, in which the target object moved, it moved a designated distance for the four remaining seconds upon recognition. In Slow Moving, the object moved for 1/3 of the frame on the test device with a resolution of 1280 × 800 pixels. In Fast Moving, the target object moved from one end of the frame to the other in the test device.

Figure 7 shows the average and median FPS of non-OFSA and OFSA observed in the same 5-s image length on a graph. As shown in Figure 7 for Stable, OFSA is tracking and rendering the target object through recognizing the interval [1–1.5 s]. Additionally, in the next step, OFSA can keep the same FPS, prior to the target object without the tracking and rendering step. For Slow Moving in Figure 7, the target object should be placed from the camera at a greater distance in Stable. Therefore, it can speed up the time more than Stable by using a few of polygons without a rendering step. OFSA experiences a small frame drop because of the process for storing snapshots compared to non-OFSA in [1–1.5 s]. Furthermore, it can keep the initial FPS in [1.5–3 s], which does not exceed the threshold value of OFSA regarding the moving of the target object. For Fast Moving in Figure 7, OFSA has to save each new frame of the snapshot within the threshold value of the moving target object for each frame. Therefore, non-OFSA has a large value of frame drop for the total interval in order to solve the adaptive operation.

The average and median FPS in Table 2 are the average and median value of 10 FPS, which were measured in 0.5-s units in a 5-s image.

$$Performance\ Sacle = \frac{FPS_{OFSA}}{FPS\ _{nonOFSA}} \times 100 \tag{4}$$

Table 2 exhibits the relative improvement rate of performance.

Table 2. Comparison of the frame rate between OFSA and non-OFSA.

Video Type	non-OFSA		OFSA		Performance Scale
	Average FPS	Median FPS	Average FPS	Median FPS	
Stable	12.8	8	26	34	203%
Slow Moving	18.2	16	22.4	30	123%
Fast Moving	13.6	10	11.6	6	85%

Equation (4) calculates the performance scale, where FPS_OFSA and FPS_nonOFSA refer to average FPS. Table 2 explains that OFSA can secure more frames than non-OFSA under the conditions of Stable and Slow Moving. In particular, it showed more than twice the improvement of performance compared to non-OFSA with greater frame recognition, under a condition of Stable, in which the target object does not move. However, the performance of OFSA decreased to 85% in the Fast Moving condition, which can indicate a limitation of OFSA due to the time delay that occurs during the memory read-writing process of the snapshot.

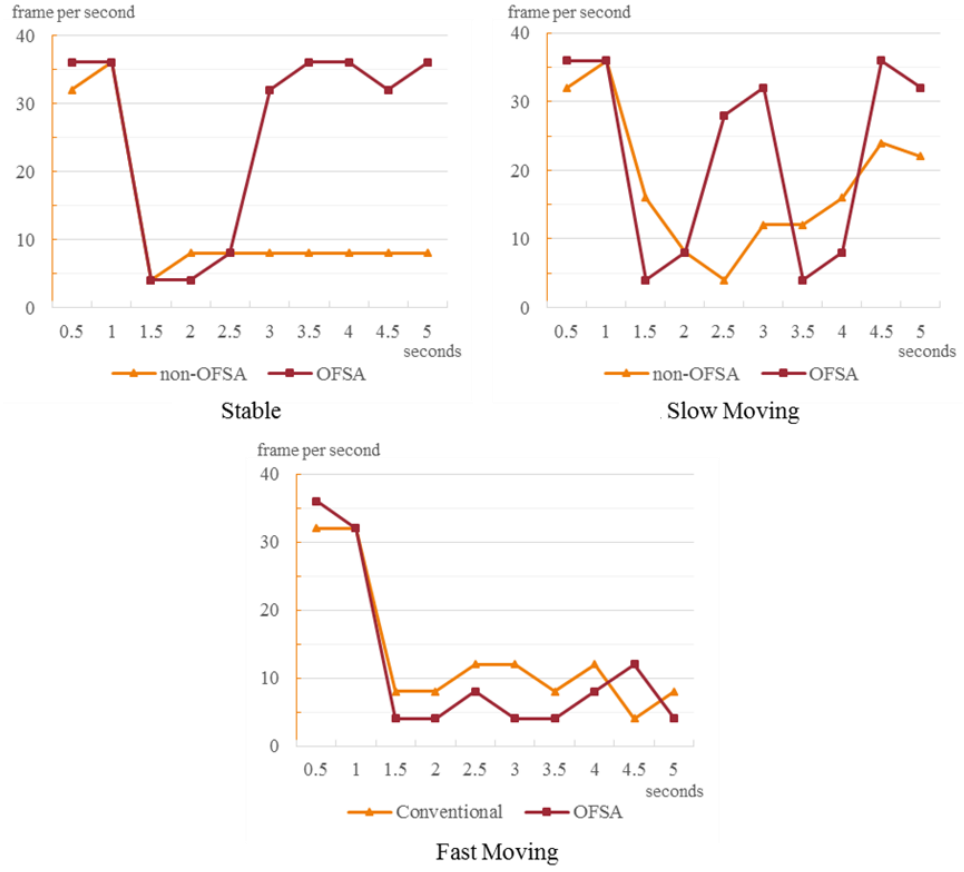

Figure 7. Comparison of the frame rate between OFSA and non-OFSA.

Figure 8 compares OFSA and adaptive OFSA using the same image. The experiment was performed to confirm that adding an adaptive operation to OFSA can achieve improved frame rates compared to OFSA in situations in which a large amount of movement occurs with the AR object. The Stable image was used in interval [0–3-s]; the Slow Moving image was used in interval [3–6-s]; and the Fast Moving image was used in interval [6–9-s]. Adaptive OFSA showed improved frame rates compared to OFSA, as it minimized registering key frames in the Fast Moving condition. As the snapshot is registered in one frame before beginning the adaptive operation, it cannot instantly respond in the Fast Moving situation. Nevertheless, adaptive OFSA showed a greater than 14% improvement of the average FPS.

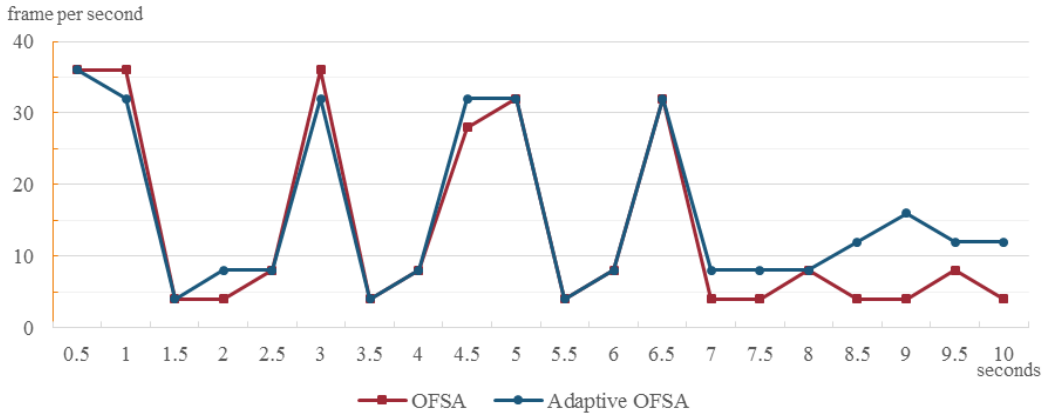

Figure 8. Comparison of the frame-rate between OFSA and adaptive OFSA.

OFSA can support a fast speed for AR services with large-scale data processing. For example, we can reduce the data processing time of server AR services in an exhibition hall, which has low computing devices, such as smart glass, smartphones and showcases, by displaying the processed data from server. Therefore, OFSA can have a minimized cost for the scale and operation of the server to AR services.

7. Conclusions

The OFSA, in this paper, is capable of providing the same adaptive AR service as that in heterogeneous adaptive symmetric services in smart device environments. The performance evaluation, which was conducted by dividing the situations into Stable, Slow Motion and Fast Moving conditions, confirmed higher frame rates compared to the existing rate in the actual AR environment. OFSA showed particularly strong performance in searching for moving targets or in entering additional target objects in a stable AR object environment. As it minimizes system resource waste due to rendering work in the case of small wobbles, it not only achieves faster tracking of target objects, but also enhances the battery life-cycle of smart devices. We believe that this algorithm can be extended, with minimum usage of system resources, to implement an AR that responds to the inevitable hand trembling that occurs due to users' ability to hold smart devices with one hand.

As the adaptive OFSA did not achieve better performance than non-OFSA in cases of target objects that constantly move at high speed, we plan to further develop this scheme in the future by using a method of storing only the valid domain in memory, to minimize the memory read-writing process.

Acknowledgments: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2053564). Furthermore, this research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-H8501-16-1014) supervised by the IITP (Institute for Information & communications Technology Promotion).

Author Contributions: All the authors contributed equally to this work. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, X.; Ren, F. Mechanism and Methods of Outdoor AR Spatial Information Visualization Representation. In Proceedings of the Second International Conference on Computer Modeling and Simulation (ICCMS '10), Hainan, China, 22–24 January 2010; pp. 272–276.
2. Wua, H.; Leeb, S.W.; Changc, H.; Liangd, J. Current status, opportunities and challenges of augmented reality in education. *Comput. Educ.* **2013**, *62*, 41–49. [CrossRef]
3. Milgram, P.; Takemura, H.; Utsumi, A.; Kishino, F. Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. *Telemanip. Telepresence Technol.* **1994**, *2351*, 282–292.
4. Klopfer, E. *Augmented Learning: Research and Design of Mobile Educational Games*; The MIT Press: London, UK, 2008.
5. Klopfer, E.; Squire, K. Environmental Detectives—The Development of an Augmented Reality Platform for Environmental Simulations. *Educ. Technol. Res. Dev.* **2007**, *56*, 203–228. [CrossRef]
6. Kobayashi, T.; Kato, H.; Yanagihara, H. Pose tracking using motion state estimation for mobile augmented reality. In Proceedings of the IEEE International Conference on Consumer Electronics (ICCE '15), Las Vegas, NV, USA, 21–23 January 2015; pp. 9–10.
7. Jang, H.-S.; Jeong, J.-Y.; Kim, Y.-H.; Yoon, Y.-J.; Ko, S.-J. Augmented reality with high frame rate for low computational power devices. In Proceedings of the IEEE International Conference on Consumer Electronics (ICCE '11), Berlin, Germany, 6–8 September 2011; pp. 274–275.
8. Comport, A.I.; Marchand, E.; Pressigout, M.; Chaumette, F. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Trans. Vis. Comput. Graph.* **2006**, *12*, 615–628. [CrossRef] [PubMed]

9. Basori, A.H.; Afif, F.N.; Almazyad, A.S.; AbuJabal, H.A.S.; Rehman, A.; Alkawaz, M.H. Fast Markerless Tracking for Augmented Reality in Planar Environment. *3D Res.* **2015**, *6*, 1–11. [CrossRef]
10. Yii, W.; Li, W.H.; Drummond, T. Distributed visual processing for augmented reality. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR '12), Atlanta, GA, USA, 5–8 November 2012; pp. 5–8.
11. Huang, B.; Lin, C.H.; Lee, C. Mobile augmented reality based on cloud computing. In Proceedings of the International Conference on Anti-Counterfeiting, Security and Identification (ASID '12), Taipei, Taiwan, 24–26 August 2012; pp. 1–5.
12. Vanus, J.; Kucera, P.; Martinek, R.; Koziorek, J. Development and testing of a visualization application software, implemented with wireless control system in smart home care. *Hum. Centric Comput. Inf. Sci.* **2014**, *4*, 18. [CrossRef]
13. Hoecker, M.; Kunze, M. An on-demand scaling stereoscopic 3D video streaming service in the cloud. *J. Cloud Comput.* **2013**, *2*, 1–10. [CrossRef]
14. Kang, H.-S. A Real-Time Integrated Hierarchical Temporal Memory Network for the Real-Time Continuous Multi-Interval Prediction of Data Streams. *J. Inf. Process. Syst.* **2015**, *11*, 39–56.
15. Degefa, F.B.; Won, D. Extended Key Management Scheme for Dynamic Group in Multi-cast Communication. *J. Converg.* **2013**, *4*, 7–13.
16. Knoche, H.; McCarthy, J.; Sasse, M.A. Can small be beautiful?: Assessing image resolution requirements for mobile TV. In Proceedings of the 13th Annual ACM International Conference on Multimedia (MM '05), Singapore, 6–12 November 2005; pp. 829–838.
17. Spencer, L.; Jakobsen, M.; Shah, S.; Cairns, G. Minimum required angular resolution of smartphone displays for the human visual system. *J. Soc. Inf. Display* **2013**, *21*, 352–360. [CrossRef]
18. API Reference of Vuforia SDK. Available online: https://developer.vuforia.com/resources/api/unity/index (accessed on 26 November 2015).
19. Rekimoto, J. Matrix: A realtime object identification and registration method for augmented reality. In Proceedings of the Asia Pacific Computer Human Interaction (APCHI '98), Kanagawa, Japan, 15–17 July 1998; pp. 63–68.
20. Bostanci, E.; Kanwal, N.; Clark, A.F. Augmented reality applications for cultural heritage using Kinect. *Hum. Centric Comput. Inf. Sci.* **2015**, *5*. [CrossRef]