

Study on AR Application Efficiency of Selected iOS and Android OS Mobile Devices

Monika Zamłyńska¹, Adrian Lasota², Grzegorz Debita¹^a and Przemysław Falkowski-Gilski²^b

¹*Faculty of Management, General Tadeusz Kosciuszko Military University of Land Forces,
Czajkowskiego 109, 51-147 Wrocław, Poland*

²*Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology,
Narutowicza 11/12, Gdansk, Poland*

Keywords: Android OS, HCI (Human-Computer Interface), Mobile Devices, Multimedia Content, UX (User Experience).

Abstract: Currently, the number of scenarios for using AR (Augmented Reality) tools grows every year. Yet, there are still some open performance issues associated with this technology, related with the efficiency of the mobile device itself, including build-in components or the operation system. The purpose of this study was to investigate available augmented reality engines for mobile platforms. First, a review of current graphics and AR engines was conducted, based on which measurement aspects were selected taking into consideration the device performance. Next, we have performed a series of research studies, including different 3-dimensional AR modes and scenarios, both with and without a tag. The research was carried out using 4 mobile devices, with 3 of them running Android OS and 1 powered by iOS (with 2 different AR libraries). The performed tests and obtained results can aid any interested individual when choosing the right mobile device, as well as configuring the AR environment, for various UX (User Experiences).

1 INTRODUCTION

Augmented reality (AR) is a strongly growing technology. To confirm this claim, according to the International Data Corporation report (IDC, 2022), in 2016 the global AR market was valued at USD 200 million, while four years later stood at USD 4.16 billion. Moreover, by 2028 its size is expected to reach USD 97.76 billion, exhibiting an excellent compound annual growth rate (CAGR) of 48.6% during the forecast period. The great players of the mobile platforms market, including Google with its Android system and Apple with iOS, either individually or in cooperation with others, are actively developing AR solutions.

Augmented reality can be defined as a system that performs three basic functions (Wu et al., 2013):

- connecting the real world with the virtual world,
- real-time interaction,
- and registration of 3D virtual and real objects.

AR simplifies our life by providing virtual information not only for the user's immediate environment, but also for any view of the real world, like a live and on-line video broadcasting (Fuhrt, 2011). Augmented reality differs from virtual reality in that AR alters and expands perceptions of the real world, where virtual reality fully transforms the user's world with the generated one (Steuert, 1992).

In other words, AR technology extends reality by superimposing virtual objects in real-time. It outlines a superior way of presenting data, that is, hiding reality from the user (Azuma et al., 2001). Removing objects from the real world corresponds to covering objects consistent with the background as information to the human that the object is not here. Augmented reality has a wide range of applications, including medical visualization, advertising, maintenance, repair, and annotation.

^a  <https://orcid.org/0000-0003-1984-4740>

^b  <https://orcid.org/0000-0001-8920-6969>

2 AR SYSTEM ARCHITECTURE

From an architectural point of view, the flow of information in an augmented reality-enabled application involves several independent modules, as shown in Figure 1:

- The physical module of the RGB camera receives a stream of reflected light from the real world to record the environment. The output from this module is a video data stream.
- This stream must be processed by a module that converts it into individual image frames.
- A marker or point cloud (depending on how the environment is recorded) is extracted from a single image frame, which will allow another module to track that object. This module has defined transformation matrices for the coordinates of the world, the camera and the object to be tracked. Thus, it is able to calculate the expected values of the future 3D object.
- The tracking module continuously transmits the calculated transformations and positions needed for rendering 3D objects.
- The final step is to display on the screen of the smartphone the image from the camera connected to the generated virtual object.

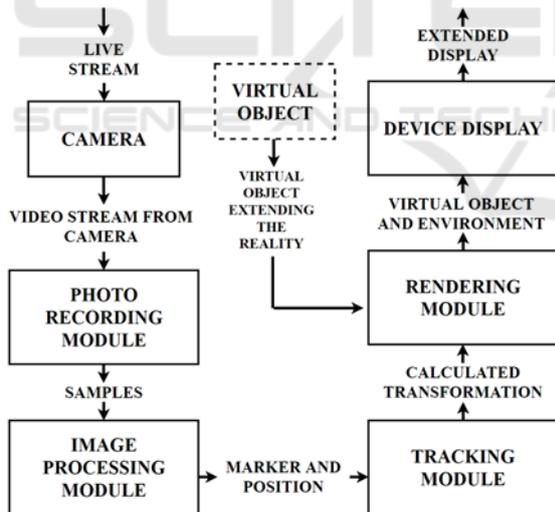


Figure 1: AR data flowchart.

Augmented reality libraries usually implement only a module related to image processing, i.e., extraction of information about markers or the ambient point cloud, and a module responsible for updating this information in real-time. Developers strive to provide the best possible experience when using AR applications, so the other modules are also included in the code they provide. As an example,

let us consider the Google’s Depth API (Android Developer, 2022), which uses spatial vision (depth sensor) in a growing number of smartphones to place virtual objects behind real things.

Figure 2 shows a comparison of scenes where, in the left image, the virtual cat is simply placed in space, while in the right image, the cat is hidden behind a plush piece of furniture.

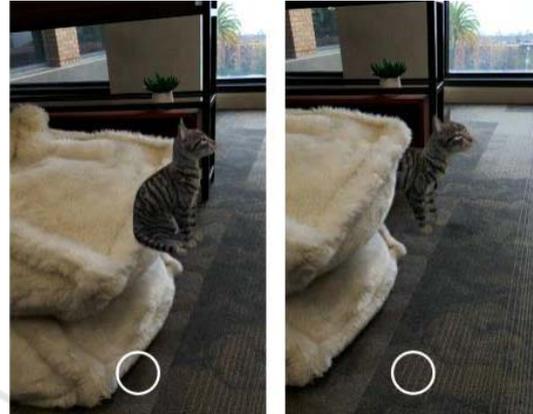


Figure 2: AR scene without (left) and with (right) applied depth (Android Developer, 2022).

Figure 3 indicates the heat map from the depth sensor, with red for a small distance from the camera, and blue for a large distance.

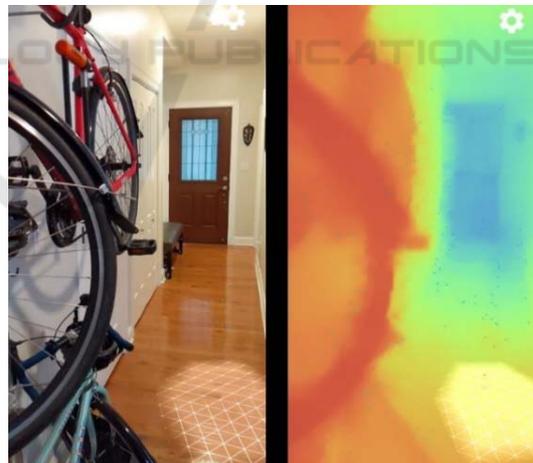


Figure 3: Camera view using a RGB (left) and depth (right) sensor (Android Developer, 2022).

3 METHODS FOR RECORDING THE ENVIRONMENT

There are two ways of recording the environment, namely marker-based or marker-less.

3.1 Marker-based

Augmented reality can be classified by the type of real world scanning. The first class is AR (marker-based) support, as shown in Figure 4.



Figure 4: Exemplary AR markers.

Markers are usually black and white due to the fact that such colors are more contrasting, allowing the camera to better register and later extract the marker from the image. This relieves the device of potential image post-production. This is not a requirement, however, and with proper lighting, color markers work without problems, but it can increase tag detection time.

The example AR tags differ in appearance, and this is due to the use of other algorithms to detect the pattern on them. The form of the tags is limited only by the implemented way of detecting it in the registered images.

When working with marker-based augmented reality, the tag must be “unpacked” by projection. Projection is a type of three-dimensional transformation, also known as perspectival transformation, or as homography operates on homogeneous coordinates.

$$\begin{bmatrix} hx_c \\ hy_c \\ 1 \end{bmatrix} = H \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} N_{11} & N_{12} & N_{13} \\ N_{21} & N_{22} & N_{23} \\ N_{31} & N_{32} & 1 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} \quad (1)$$

where: H is an arbitrary 3 x 3 homogeneous matrix, x_{ci} , y_{ci} , for $i = 1, 2, 3, 4$, are the coordinates read (marker), x_{mi} , y_{mi} , for $i = 1, 2, 3, 4$, are the expected coordinates (real world).

After putting the resulting values and equating all sides of the equation, we get a transformation matrix, which we use to transform 3D objects to place it in real space. Projection is commonly used for scanning documents through smartphones or transforming photos where there are lens distortions.

The tag has a specific layout (Gupta, 2019), as shown in Figure 5, to orient it in space. The tag-

recognizing algorithm contains the coordinates x_{mi} , y_{mi} for each feature point on the tag, comparing this with the coordinates read from the photo through homography, from which we receive the desired transformation.

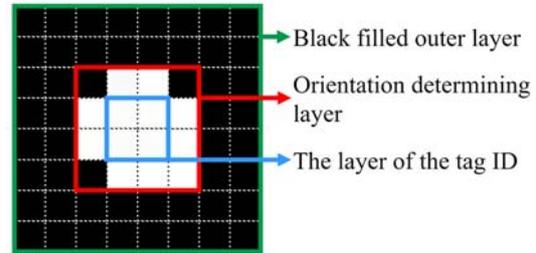


Figure 5: Layout of an exemplary tag.

3.2 Marker-less

The second class is marker-less operation. The advantage of this mode of operation is the ability to move the phone in space without having to aim the marker. Proper marker-less operation requires flat ambient surfaces, preferably distinguishable from the background.

Surface recognition proceeds by mapping the environment. Depending on the algorithm implemented in a library, individual image frames are analyzed using feature extraction technology or simultaneous localization and mapping (SLAM) technology, among others, to search for correlations between frames. Once connections are detected on a certain number of frames, 3D points are defined based on a point cloud, also known from photogrammetry.

SLAM's purpose is to move through space, scanning it to create a digital map of the space for the user to refer to. Initially, the technology was used to help robots move without colliding with their surroundings, but it is now appearing in many other industries as well (Kolhatkar and Wagle, 2021). There are two types of SLAM:

- visual SLAM (VSLAM), which is based on scanning through a camera,
- and light detection ranging (LiDAR), which uses laser scanners to analyze the environment.

Both types of SLAM often use inertial measurement unit (IMU) support, which can be a gyroscope, accelerometer, Hall sensor, etc. This makes it possible to achieve better maps. When the camera captures the surroundings, the IMU tracks the device's movement, allowing it to measure the distance between any two images, improving the aspect ratio of the digital map.

4 GRAPHICS ENGINES

To run augmented reality on any smartphone, one must first be able to display additional digital information on the screen. The software that communicates with the device's graphics card through its drivers is responsible for handling the digital world. The simplest way to display custom objects is to use the graphics engine or more low-level API, which defacto is used in the graphics engine anyway. Examples of graphics APIs include: Vulkan, OpenGL, Direct3D or WebGL, which is an API for drawing in a web browser as part of a JavaScript extension.

A graphics engine is at a higher level of abstraction than a graphics API. It uses the API to render an image, but provides all the tools needed to create, e.g., a computer game. The list of tools includes, among others, an engine that simulates physics or performs collision detection and accompanying mechanisms, support for sound, scripting, animation, memory management, network support and more. The list of available graphics engines is large, as most companies on the global game market have proprietary engines, such as: CryEngine from Crytek, id Tech from id Software, or REDEngine from CD Projekt RED. Not all of them are compatible with mobile devices. On smartphones, the main ones used are: Libgdx, Godot, ShiVa3d, Marmalade SDK and CocoonJS, but none of them support augmented reality. Engines with AR support include Unity and Unreal Engine.

4.1 Unreal Engine

Unreal Engine written in C++ is produced by Epic Games, the engine received its fifth release in 2022. This language is also used to write scripts since version 4, previously it was the proprietary UnityScript language. A parallel way of writing scripts is Blueprints, a visual language based on blocks and flows. It is used to handle level events, control the behavior of actors and for complex animations through a highly realistic character system. Examples of the use of games using this engine are: Fortnite, the Gears of War series or Mass Effect 3.

The engine is widely used not only in video games but also in film productions, as exemplified by Project Spotlight. One of the presentation parts of the project was a demonstration of the engine's capabilities within real-time technology. Epic Games, together with Lux Machina, Magnopus, Profile Studios, Quixel, ARRI and DP Matt Workman,

demonstrated how LED walls can provide not only virtual environments but also lighting for real-world elements. This allows quick modification of scenes, time of day and lighting. Control is done via virtual reality workstations, tablets or APIs (Epic Games, 2022).

4.2 Unity

Unity has existed since 2005 as a product of Unity Software Inc. (business-wise Unity Technologies). The engine is written in C++, while the scripting language is C#. Additional extensions allow scripts to be written visually, similar to Unreal Engine. Productions based on Unity are: League of Legends, Cuphead or Ori. Unity has native support for AR libraries (Unity, 2021).

The support provides common interfeatures using the various libraries underneath. This allows for unified production on Android and iOS. In 2020, Unity Technologies released a studio that builds on MARS augmented reality from the ground up. This introduces, among other things, axes reflecting the real world (centimeters instead of pixels), determining conditions based on distance or viewing angles. The program provides a smartphone application for dynamic programming, a simulation of the real environment, along with a device simulation and a hypothetical process for mapping the environment.

5 AR LIBRARY

The AR library is a collection of tools needed to support augmented reality. This includes tracking the movement of the device, recognizing the environment (detecting the size and location of any type of surface, vertical, horizontal and angled) and estimating illumination from the environment.

The most popular libraries for AR are Google's ARCore and Apple's ARKit, so these are the focus. Both libraries have plugins to support usage in the aforementioned graphics engines. A strongly developing library is Vuforia Engine, while it requires a paid license for commercial purposes.

5.1 ARKit

In 2017, Apple released a collection of tools working with virtual and augmented reality. The library was one of the components of the demonstration of the new iOS 11 system presented at WorldWide Developers Conference WWDC'17 (Apple, 2017).

The library's main functionalities are: enhanced face tracking, placing objects in specific geometric locations, detailed recognition of the environment (walls, windows, doors, seats), capturing the movement of people (bones and joints), penetrating through people, and simultaneous operation on the front and rear cameras (Apple Developer, 2022).

5.2 ARCore

The first work on Google's AR library began in 2012 with the emergence of the Tango project, advertised as "technology that gives devices the ability to interact with the real world as we do as humans". In 2018, Google announced the extinction of the project in favor of ARCore, which is a continuation of AR development (Kastrenakes, 2017). The functionalities are very similar compared to ARKit. The advantage of the library is support for the competitor's operating system, namely iOS. ARCore is supported by both Android and Apple devices.

6 RESEARCH PROBLEM

Since the two libraries, that is ARKit and ARCore, have similar functionalities, the main aim was to evaluate the effectiveness of both solutions in a similar test scenario.

6.1 Utilized Hardware and Software

To prepare the testbed, a PC with the following specification was used:

- 8-core 16-thread Intel Core i9-9900k processor clocked at 4.68 GHz, 16 GB of DDR3 RAM clocked at 3 200 MHz and a SATA SSD, Windows 10 Education 20H2 operating system,

and a second PC with:

- 6-core 12-thread Intel Core i7-8700B processor clocked at 3.6 GHz, 16 GB of DDR3 RAM clocked at 3 200 MHz, and a SATA SSD, macOS Big Sur 11.5 operating system.

This information is necessary in case of investigation of the application compilation time.

The chosen graphics engine was Unity version 2020.3.3f1, due to having more experience in this ecosystem. The C# programming environment used (for scripting) was JetBrains Rider 2020.

The tested mobile devices included:

- Samsung S9+ smartphone with Android 10,
- Samsung A10 smartphone with Android 9,
- Lenovo Tab M10 Plus tablet with Android 9,
- Apple iPhone X smartphone with iOS 14.

Principle technical specification of each smartphone and tablet are described in Table 1.

Table 1: Technical specification of tested mobile devices.

Device	Component and description
Samsung S9+	8-core 2.9 GHz CPU, Mali-G72 MP18 GPU, 6 GB RAM, 6.2 inch 1440x2960 super AMOLED display, 3500 mAh battery
Samsung A10	8-core 1.6 GHz CPU, Mali-G71 MP2 GPU, 2 GB RAM, 6.2 inch 720x1520 IPS TFT display, 3400 mAh battery
Lenovo Tab M10 Plus	8-core 2.3 GHz CPU, PowerVR GE8320 GPU, 4 GB RAM, 10.3 inch 1200x1920 IPS TFT display, 5000 mAh battery
Apple iPhone X	6-core 2.39 GHz CPU, Apple GPU, 3 GB RAM, 5.8 inch 1125x2436 OLED display, 2716 mAh battery

As shown, they came from different manufacturers and include both operating systems, namely Android and iOS, with Android available in version 9 and 10.

6.2 Research Method

The research began with an analysis of indicators to draw conclusions. There are numerous studies on benchmarking in many industries (Dai and Berleant, 2019) and many characteristics of these studies can be applied to mobile devices (Kim and Kim, 2012; Patton and McGuinness, 2014; Hirsch et al., 2021). The study on gaming performance index presents 5 groups of measurement aspects (Dar et al., 2019):

- Visual fluidity.
- Temperature.
- Battery life.
- Responsiveness.
- Graphics.

All of them were taken into account in our experiments.

6.3 Measurement Apparatus

Visual smoothness refers to the number of frames per second of an application. Poor smoothness is referred to by users in a number of ways, including: laggy, not smooth, with significant delay, etc. When the frame rate is too low, the illusion of smooth animation movements disappears and the user notices individual images. An unstable frame rate can cause abnormal response and action movements by the user. A standard set of smoothness parameters are: average frame rate, percentile score and frame stability.

Temperature is a major issue for limited performance. Compared to PCs, where typically the graphics card and processor operate at the highest frequency at a constant temperature (higher cooling efficiency), smartphones are more prone to temperature increases due to the scale (small form factor). High device temperature directly affects the comfort of handling the device, so it also influences the time of use without interruption. Therefore, temperature is a key measurement component.

Currently on the market, smartphones have larger and larger batteries. However, when, e.g., continuously processing demanding audio-visual data, it settles at an average of 7 hours, so the amount of power consumed by the application is very important. The battery indicator can be broken down into the degree of charging and discharging of the device during use.

Libraries have different implementations of ambient recognition, which affects the time it takes to display 3D objects. Responsiveness is the time it takes to act based on user decisions, e.g., changing the displayed objects. When using 3D graphics, we were able to evaluate the performance of the same game engine on two operating systems. Subjective evaluation will be given to the quality of textures, shadows, smoothing, and quality of effects.

6.4 Method of Measurement

In Unity, thanks to appropriate extensions for the component responsible for testing, called Unity Test Runner, it was possible to carry out performance tests (Unity, 2022). The extension API has several groups of methods:

- Measure.Method – executes the indicated method measuring performance.
- Measure.Frames – allows to record metrics based on individual frames.
- Measure.Scope(...) – allows to record metrics within a specified range (e.g., 3D object).

- Measure.FrameTimes(...) – captures metrics based on frame time (milliseconds).
- Measure.ProfilerMarkers(...) – records metrics for specific markers (time-based or marked in script code).
- Measure.Custom(...) – allows to record metrics based on other reference points than methods, frames, and frame time.

Parameters that Unity is unable to provide, such as battery level and device temperature, will be read out using an application, namely Sensors Multitool, which will register individual sensors in the background. After testing, the logs will be processed, analyzed and compared for all devices.

7 AR APPLICATION

For the purpose of this study, we have developed our own custom-build AR mobile application.

7.1 Configuration

Two scenes implementing the use of both marker-based and marker-less approaches were prepared for the analysis. To start working with augmented reality in Unity, one needs to activate or install the AR Foundation package (Unity, 2021), which allows to work on multiple platforms (including Android and iOS) at once. Subsystems are included in other packages: ARKit XR Plugin and ARCore XR Plugin. All packages are available within the Unity Package Manager. After installing the package, it was necessary to set the part that manages the interaction with the world, i.e., XR Management.

The core component of the scene is the AR Session component, which controls the lifecycle of the AR experience by enabling and disabling augmented reality on the device. This component is configured globally, so by defining a session multiple times, it will manage the shared session. One configurable option is “Attempt Update”, attempting to install the required AR software on the device if necessary and possible.

The second component is the AR Session Origin. It stores the camera object and all objects noticed in the detection process (markers or point clouds). The component converts the transformation parameters (position, orientation and scale) of the found elements to the correct values for the final Unity space.

The camera object is a regular camera used in Unity with AR scripts attached to it. The most important of these is the AR Pose Driver that controls

the position and orientation of the parent object (in this case, the camera object) with respect to information from the device. Another element bundled with the camera is the AR Camera Manager enabling AR functions, such as light estimation from the real environment with texture control.

7.2 Surface Recognition

The component responsible for object recognition is called Trackable Manager. Thanks to it, it is possible to detect various types of objects. AR Foundation supports recognition of various elements:

- ARPlaneManager – detects flat surfaces,
- ARPointCloudManager – detects key points in the form of a cloud,
- ARAnchorManager – manages nodes, allows to add and remove custom points in space,
- ARTrackedImageManager – detects and tracks 2D images,
- AREnvironmentProbeManager – a technique that allows to capture the environment as a texture, e.g., to represent realistic reflections on an AR object,
- ARFaceManager – detects and tracks human faces,
- ARTrackedObjectManager – detects 3D objects,
- ARParticipantManager – detects other users in case of group operations.

In the described research, we will focus on 2D image tracking (markers) and surface detection (marker-less).

7.3 AR Tracked Image Manager

The mechanism of operation of image recognition is to detect predefined data and track it, it forces to prepare images to serve as AR tags. A library available from the AR Foundation, called Reference Image Library, can help with this.

Completing the library is done by adding a new image and filling in the required fields. One of them is a link to an image file in a Unity-compatible format. This file can be digitally prepared, then printed, or as a product of transferring a physical object into digital space through a scan or photo. It is possible to define the name of the image, and set the physical size of the marker to help with later transformation. Then, after defining the library, one can point to it in the tracking manager as Serialized Library.

At this point, there are several ways to place AR objects in reality, one of which is to indicate a 3D object as a Tracked Image Prefab, by which Unity,

upon detecting one of the images from our library, will automatically add the object in question as pinned to the tracked image.

Another option is to do it in a script using the “ScheduleAddImageWithValidationJob” method called on our library. The method takes the same parameters as the definition in the manager, but this allows to dynamically change the images, i.e., through user interaction.

7.4 AR Plane Manager

Plane manager deals with storing and modifying found groups of points defined as a plane. One of the arguments it takes is Plane Prefab, the object that will represent the found area. The available default object is a black line that is the outline of the surface and a slightly transparent brown fill for this space, as shown in Figure 6.



Figure 6: Example of a found area on a desk next to a computer keyboard.

Once the surface is registered, it is possible to place 3D objects on it. One way to do this is by raycasting in the direction of the surface. If such a surface consists of polygons (and not lines), the point of intersection of the ray with the surface becomes the point of attachment of the 3D object (with the appropriate transformation relative to the surface). The above process is done with a custom script.

7.5 Evaluated 3D Models

In order to test several measurement scenarios, a collection of 3D models with varying levels of detail was prepared. The collection includes models of passenger cars:

- Auto number 1 – 20 200 triangles and 35 500 vertices, 2048x2048 pixels (see Figure 7),



Figure 7: Car model no. 1.

- Auto number 2 – 23 627 triangles and 14 499 vertices, 512x512 pixels (see Figure 8),



Figure 8: Car model no. 2.

- Auto number 3 – 137 300 triangles and 74 800 vertices, 4096x4096 pixels (see Figure 9),



Figure 9: Car model no. 3.

- Auto number 4 – 1 200 000 triangles and 632 000 vertices, 2048x2048 pixels (see Figure 10).

The cars were placed on both scenes, i.e., a scene using markers and a scene using surface detection. Two tests were performed for the scenarios: displaying a single model and changing to the next



Figure 10: Car model no. 4.

one, and rendering all models gradually increasing the number of models, thus testing the speed of model swapping and the load degree.

8 RESULTS

The scenarios were prepared to eliminate the human factor that could affect measurements. Automatically, after the application starts, the time of detection of the marker or first plane is measured. Then, at 5-second intervals, the 3D model is changed to the next one. When the last model is swapped, the application stops, and the performance testing procedure ends.

The first measurement, as shown in Figure 11, is the marker search time. The values shown represent averaged results from 5 measurements.

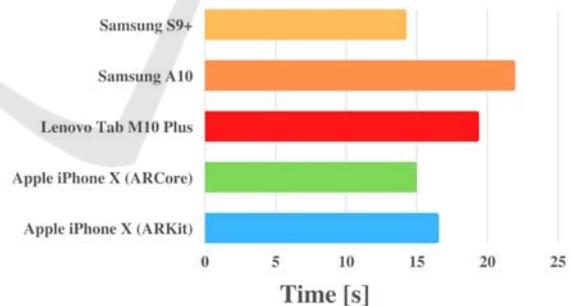


Figure 11: Results for marker search time.

Despite the fact that ARKit was released a year earlier than ARCore, the library from Google performs better on similar hardware compared to the library from Apple. On the other hand, hardware that is less powerful, namely the Lenovo Tab M10 Plus and Samsung A10, performs worse than iPhone.

The level of battery consumption was not achievable on most devices, and if it was available, it was meaningless, as the values were extremely different when inactive.

Another measurement, shown in Figure 12, was the loading time of the largest model, namely model no. 4. The average number of frames during this scenario is shown in Figure 13.

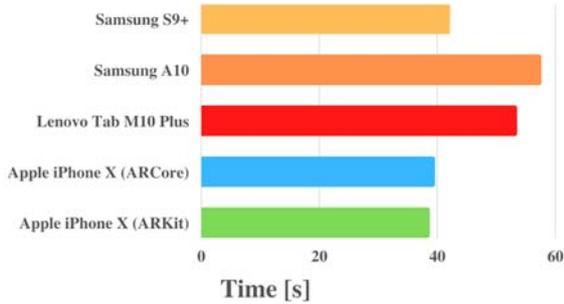


Figure 12: Results for rendering time.

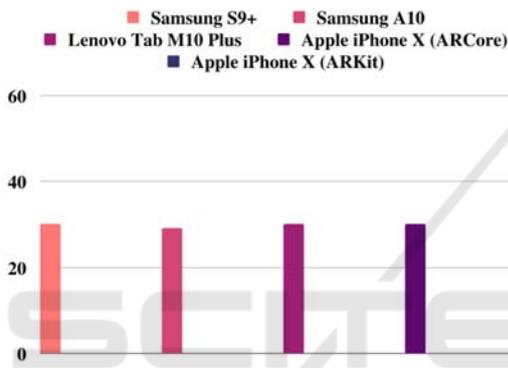


Figure 13: Results for FPS.

Currently, ARCore is limited to 30 frames per second. This shows the advantage of ARKit over Google. However, this library is only available for the Apple’s iPhone and iPad. ARCore performs a little worse on the same device, due to worse optimization on the competitor’s platform. For all devices, the score percentile was as high as the average, which means that frame stability was maintained.

The quality of 3D models was the same on all devices, due to the same settings for edge smoothing and shadow quality. In contrast, the effect of the AR function differed significantly. The estimated light for ARKit looked much better and behaved more dynamically relative to the prevailing conditions. Illuminating the scene with an additional light source natively also illuminated the model, where in ARCore the system took 2 seconds to calculate the generated light.

The next tested factor was the temperature of the device, shown in Figure 14, after running the measurement scenario 5 times.

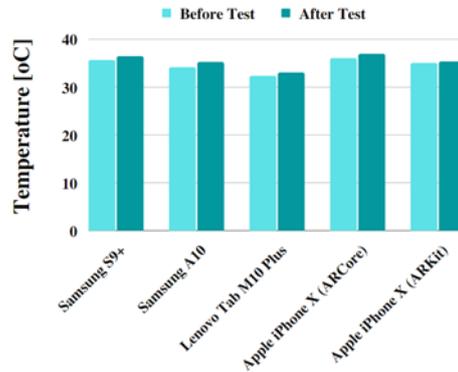


Figure 14: Results for temperature.

The lowest temperature increase was registered for the iPhone X device with the native library, while the highest for the Samsung A10. Hence, it can be concluded that ARCore is more demanding than ARKit.

9 SUMMARY

After analyzing the presented results, it can be concluded that ARKit proved to be a better library for AR than ARCore. One of the key reasons for this state of performance may be better optimization for a given pool of devices. Apple offers a smaller number of devices under its brand, while Android is available on more than 70% of devices worldwide (Statcounter, 2022). It is simply easier for Apple to fine-tune the library for its hardware. ARCore, on the other hand, is released under the Apache 2.0 license, i.e., with open source code, allowing any user to contribute to developing the library.

Future studies may and should include a wider group of devices, particularly other smartphones and tablets both from Apple and various Android-powered manufacturers. It would be also interesting to collect other common functionalities of the libraries, such as face tracking, and prepare additional scenarios for them. This would point out the pros and cons from a broader point of view.

When it comes to analyzing the temperature of tested devices, it would be surely interesting to use a thermal imaging camera. The described libraries introduce new functionalities depending on the processor, so the next step would be to analyze the behavior of both libraries on flagship Android and iOS devices. Additional source of inspiration may be found in (Xia et al., 2019; Falkowski-Gilski, 2020; Falkowski-Gilski and Uhl, 2020; Jacob et al., 2021; Lee et al., 2021).

REFERENCES

- Android Developer. (2022). <https://developers.google.com/ar/develop/depth> (access: 21.07.2022).
- Apple. (2017). Highlights from WWDC 2017. <https://www.apple.com/pl/newsroom/2017/06/highlights-from-wwdc-2017/> (access: 21.07.2022).
- Apple Developer. (2022). ARKit. <https://developer.apple.com/augmented-reality/arkit/> (access: 21.07.2022).
- Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. (2001). Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6), 34-47.
- Dai, W., Berleant, D. (2019). Benchmarking contemporary deep learning hardware and frameworks: a survey of qualitative metrics. In *CogMI'19, 2019 IEEE First International Conference on Cognitive Machine*. IEEE.
- Dar, H., Kwan, J., Liu, Y., Pantazis, O., Sharp, R. (2019). The game performance index for mobile phones. arXiv preprint arXiv:1910.13872.
- Epic Games. (2022). Unreal Engine. <https://www.unrealengine.com/en-US/> (access: 21.07.2022).
- Falkowski-Gilski, P. (2020). On the consumption of multimedia content using mobile devices: a year to year user case study. *Archives of Acoustics*, 45(2), 321-328.
- Falkowski-Gilski, P., Uhl, T. (2020). Current trends in consumption of multimedia content using online streaming platforms: a user-centric survey. *Computer Science Review*, 37, 100268.
- Furht, B. (2011) *Handbook of Augmented Reality*, Springer. New York.
- Gupta, S. (2019). Custom AR tag detection, tracking and manipulation. <https://medium.com/lifeandtech/custom-ar-tag-detection-tracking-manipulation-d7543b8569ea> (access: 21.07.2022).
- Hirsch, M., Mateos, C., Zunino, A., Toloza, J. (2021). A platform for automating battery-driven batch benchmarking and profiling of Android-based mobile devices. *Simulation Modelling Practice and Theory*, 109, 102266.
- International Data Corporation. (2022). Press Releases. <https://www.idc.com/> (access: 21.07.2022).
- Jacob, I. J., Shanmugam, S. K., Piramuthu, S., Falkowski-Gilski, P. (eds.). (2021). *Data Intelligence and Cognitive Informatics. Proceedings of ICDICI 2020*, Springer. Singapore.
- Kastrenakes, J. (2017). Google's Project Tango is shutting down because ARCore is already here. <https://www.theverge.com/2017/12/15/16782556/project-tango-google-shutting-down-arcore-augmented-reality> (access: 21.07.2022).
- Kim, J. M., Kim, J. S. (2012). AndroBench: Benchmarking the storage performance of Android-based mobile devices. In Sambath, S., Zhu, E. (eds) *Frontiers in Computer Education*, Springer, Berlin, 667-674.
- Kolhatkar, C., Wagle, K. (2021). Review of SLAM algorithms for indoor mobile robot with LIDAR and RGB-D camera technology. In Favorskaya, M. N., Mekhilef, S., Pandey, R. K., Singh, N. (eds). *Innovations in Electrical and Electronic Engineering*, Springer, Singapore, 397-409.
- Lee, J., Wang, P., Xu, R., Dasari, V., Weston, N., Li, Y., Bagchi, S., Chaterji, S. (2021). Benchmarking video object detection systems on embedded devices under resource contention. In *EMDL'21, 5th International Workshop on Embedded and Mobile Deep Learning*. ACM.
- Patton, E. W., McGuinness, D. L. (2014). A power consumption benchmark for reasoners on mobile devices. In Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Noy, N., Groth, P., Janowicz, K., Goble, C. (eds.). *The Semantic Web – ISWC 2014*, Springer, Cham, 409-424.
- Statcounter (2022). Mobile operating system market share worldwide. <https://gs.statcounter.com/os-market-share/mobile/worldwide> (access: 21.07.2022).
- Steuer, J. (1992). Defining virtual reality: Dimensions determining telepresence. *Journal of Communication*, 42(4), 73-93.
- Unity. (2021). About AR Foundation. <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html> (access: 21.07.2022).
- Unity. (2022). Performance testing extension for Unity test runner. <https://docs.unity3d.com/Packages/com.unity.test-framework.performance@0.1/manual/index.html> (access: 21.07.2022).
- Wu, H. K., Lee, S. W. Y., Chang, H. Y., Liang, J. C. (2013). Current status, opportunities and challenges of augmented reality in education. *Computers & Education*, 62, 41-49.
- Xia, C., Zhao, J., Cui, H., Feng, X., Xue, J. (2019). Dnntune: automatic benchmarking DNN models for mobile-cloud computing. *ACM Transactions on Architecture and Code Optimization*, 16(4), 1-26.