COMPARISON OF SELECTED AUGMENTED REALITY FRAMEWORKS FOR INTEGRATION IN GEOGRAPHIC CITIZEN SCIENCE PROJECTS

C. Berger [1]*, M. Gerke[1]

[1] Institute of Geodesy and Photogrammetry, Technische Universität Braunschweig, Germany – (cosima.berger, m.gerke)@tu-bs.de

Commission IV, WG IV/4

KEY WORDS: augmented reality, geographic citizen science, mobile application, mobile development, AR implementation, location-based tracking

ABSTRACT:

Augmented reality (AR) offers functionalities that can be beneficial for citizen science (CS) projects. Especially location-based approaches have potential for geographically oriented CS projects, as objects can be placed based on geographic coordinates. Since choosing a suitable AR framework for integration into a CS project can be challenging, this paper gives an overview of the common AR frameworks and takes a closer look at three selected ones that are particularly suitable for CS projects (AR.js, AR Foundation, and ViroReact). Prototypes were implemented for the selected frameworks to investigate which framework is best suited for specific use cases. Marker-based tracking approaches, image recognition and location-based placement were considered. The results show that the framework AR.js is particularly suitable for marker-based tracking with very simple markers and therefore represents a good entry point for CS projects to integrate initial AR functionality into their project. AR Foundation and ViroReact, on the other hand, are faster and more reliable with the detection of more complex markers. The location-based approach can be implemented with all three frameworks, but the precision of the placement of the objects strongly depends on the accuracy of the sensors of the mobile device.

1. INTRODUCTION

Augmented reality (AR) is becoming increasingly attractive in industry, education, and science. AR aims to enhance the real world by displaying artificial objects on a digital screen. In a realistic AR solution, computer-generated objects blend perfectly into the user's environment, creating the illusion that the objects are truly tangible (Milgram and Kishino, 1994). In science, this presents new opportunities, as data can be presented in a way that is more understandable and approachable for non-professionals. Therefore, AR can be especially of interest in the field of citizen science (CS), as the concept of CS thrives on the idea of involving non-professionals into scientific projects. Thus, AR can be a valuable tool to open new possibilities in data visualisation and collection in CS projects.

With the support of mobile AR applications that can be run on regular smartphones, data already collected in a CS project can be presented to participants in an engaging and interactive way. By integrating the data as objects directly into the participant's surrounding, the participant can explore the data and view it from all sides by physically moving around it. In this way, AR can help participants to better understand complex subjects, as the data is presented in a visual and almost tangible way. But of course, the advantages of AR are not limited to data visualisation. AR can also be used to support and guide the participant of a CS project during data collection. For example, AR can be used to indicate the exact position where the participant should stand to collect new data. This can be particularly helpful for taking photos at various times that all show the same part of a landscape from the same angle (Albers et al., 2017).

AR is already being used in some CS projects, but is still in its infancy. However, some successful examples of the use of AR can be found, especially in geographically oriented CS projects (Sermet et al., 2018, Sansom et al., 2016). Geographic citizen science (GCS) projects are characterised by the fact that the collected data have additional spatial information attached to them. When using a mobile application in the project, the spatial information can be collected with the help of the integrated GPS sensor of the phone. In an AR solution, the same sensor can help to place objects at defined geographic positions by following a location-based tracking approach.

The benefits of AR in CS projects are evident, but implementing a mobile AR application is still challenging. Since many AR frameworks already exist, it is difficult to choose the one that best fits the requirements and constraints of a project. The wide range of available AR frameworks can seem overwhelming at first. However, it is important to be deliberate, as the choice of the AR framework is an important step in the implementation process and can have an impact on the overall success of the project.

This paper aims to give an overview of the currently available AR frameworks and to recommend which ones are particularly suitable for the use in GCS projects. Therefore, typical requirements for an AR framework to be used in a GCS project are firstly defined to identify the most suitable AR frameworks. Then, a small selection of promising frameworks is compared in terms of their characteristics and usability. To provide a

* Corresponding author

meaningful comparison, prototypes are implemented based on these frameworks to explore their functionality in practice. The comparison is intended to support future project managers of GCS projects in choosing a suitable AR framework.

2. RELATED WORK

Augmented reality extends the real world for a mobile user by displaying virtual, computer-generated objects on the device's display (Chatzopoulos et al., 2017). In comparison with virtual reality, the user does not completely immerse into a virtual world, but is still experiencing his real surrounding. Augmented reality only aims to enhance the world the user is already seeing and does not completely reconstitute it (Carmigniani et al. 2011). Due to the improved performance of mobile devices in recent years, AR can now be experienced on common smartphones (Hasler et al., 2020). This means that with the right mobile application, AR can be easily accessed by ordinary citizens.

A key challenge of AR is that the computer-generated objects must be placed in correspondence to the real environment. Objects should stay aligned and anchored to the real world even when the user moves the mobile device (Rabbi and Ullah, 2013). This challenge can also be observed in the context of AR used in CS projects. A simple possibility is to use either marker tracking or an image recognition approach. The mobile application recognises a marker/image, which is placed in the participant's environment, with the help of the camera and displays a predefined object at that location. If the user moves around the marker/image, the model remains anchored and can be viewed from all sides. The major disadvantage of this method is that the participant's environment must be prepared with markers/images beforehand (Chatzopoulos et al., 2017; Rabbi and Ullah, 2013), which is not always possible in CS projects. Location-based tracking, on the other hand, works via the positional sensors of the mobile device. Objects are placed in the user's environment on the basis of geographical coordinates, without the scene having to be prepared with markers in advance. This approach is therefore an attractive option for large-scale GCS projects, where a preparation of the entire area would not be feasible.

CS projects can and are already taking advantage of this possibility by integrating AR into their data collection or visualisation process. Particularly in GCS projects, interesting opportunities can be identified for AR to enhance the data collection. For example, Sermet et al. (2018) show that AR can be used to maintain power lines by proposing an approach where power line sag is monitored with the support of citizens who are using an AR application on their own smartphone. Another example is the *Fireballs in the Sky* application, which enables the user to recreate a fireball sighting by using the camera of their smartphone. Through the AR implementation, the user can easily pinpoint the altitude and the azimuth of the fireball sighting and send this information together with a location to the project's database (Sansom et al., 2016). The study by Albers et al. (2017) describes an approach that can be used to navigate participants of a project through an AR component to a specific location to take a photo with a predefined orientation. This approach demonstrates how AR can be used interactively to obtain photo series with well-comparable and analysable images.

Comparative studies that examine the functionalities of AR frameworks have already been conducted. Amin and Govilkar (2015) discussed the advantages and limitations of selected AR software development kits (SDKs) and predict that AR will play a greater role in the future. Other comparisons are more focused on the specific field in which the AR framework is to be used. For instance, Herpich et al. (2017) compared multiple AR frameworks concerning their usability in educational applications. Closely related to the topic of the paper is the comparison by Burkard et al. (2007), who examine existing AR frameworks regarding their usability in location-based mobile applications. However, a comparison of AR frameworks with the intention of using one in a GCS projects was not found during the research for this paper.

3. METHODOLOGY

3.1 Requirements towards an AR framework

Before implementing a mobile application, it should be determined which functional and non-functional requirements are imposed on it. The functional requirements describe which functionalities should be included in a software product, while the non-functional requirements describe the performance or quality with which these functionalities should be achieved. For example, a functional requirement for a mobile AR application could be that a marker should be recognised and as a result a 3D object is rendered in the scene. A non-functional requirement, on the other hand, would be that the marker should be detected within 3 seconds, regardless of the used mobile device. The functional requirements for a mobile application to be used in a (G)CS project will vary greatly depending on the project. This depends on the activities the user is to perform with the app. For the non-functional requirements, it is easier to identify similarities between distinct projects.

For instance, the majority of projects aim to attract a large and diverse group of participants (Wiggins, 2013). This is only possible if the mobile application integrated into the project can be used on a wide range of smartphones. An application that supports only one operating system will exclude a sizable group of potential participants. A mobile application that is used in a CS project should therefore preferably support all of the common platforms (Android, iOS, and possibly also Windows). In addition, costs often play a major role in CS projects. In particular, small CS projects may only have a limited budget, so the choice of a development framework should not generate any additional costs. Open source products offer an attractive alternative to commercial products, since apart from the cost factor, the code is also freely accessible and can be adapted and extended specifically for the project. However, open source projects always carry the risk that maintenance will be discontinued and that additional resources and expertise are required during implementation (Wiggins, 2013).

This paper focuses on AR frameworks that are open source or freely usable, despite the limitations of open source projects mentioned above. Additionally, only AR frameworks that enable cross-platform implementation are considered. Of course, this is only a very small selection of requirements, but since these two can be identified in many CS projects, it is a good basis for a first comparison.

3.2 Available AR frameworks

A selection of currently available AR frameworks is shown in Table 1. It lists open source, freely available and proprietary frameworks. For the open source solutions, all source code is

Name	Provider	Open Source	Platforms	Website / Documentation
ARCore	Google	Yes (Apache License)	Android, iOS (limited)	https://developers.google.com/ar
AR Foundation	Unity	No (Unity Companion License)	Android, iOS	https://unity.com/unity/features/arfoundation
AR.js	AR.js Org Community	Yes (MIT License)	all phones with WebGL and WebRTC	https://ar-js-org.github.io/AR.js-Docs/
ARKit	Apple Inc.	No	iOS, iPadOS	https://developer.apple.com/augmented-reality/arkit/
ARToolKit / artoolkitX	artoolkitX team	Yes (GNU Lesser General Public License version 3.0)	Android, iOS, Linux, macOS, Windows	http://www.artoolkitx.org/
Kudan AR SDK	XLsoft Corporation	No	Android, iOS	https://www.xlsoft.com/en/products/kudan/ar-sdk.html
ViroReact	ViroMedia / Viro Community	Yes (MIT License)	Android, iOS	https://viro-community.readme.io/
Vuforia	PTC Inc.	No	Android, iOS, Lumin OS, Windows	https://www.ptc.com/en/products/vuforia
Wikitude	Wikitude GmbH	No	Android, iOS, Windows	https://www.wikitude.com/

Table 1. A selection of currently available AR frameworks / SDKs / platforms

publicly available and can be modified. Some freely available frameworks do not share the source code, but the solutions can be integrated into the own software at no additional cost. For the proprietary AR frameworks, on the other hand, a license must be purchased. Since the availability of different AR frameworks is tremendous, the list does not claim to be complete. However, it already shows which options there are regarding the choice of an AR framework and how difficult it can be to select a suitable one for a specific CS project.

Google's ARCore and Apple's ARKit SDKs are the main tools to implement AR applications on Android and iOS devices. The SDKs are already integrated in a multitude of newer devices and provide important AR functionalities like plane detection, image/motion tracking, face detection, and light estimation (Oufqir et al. 2020).

AR.js is a completely web-based solution and can be easily integrated into existing web applications (AR.js Org, 2022). ViroReact is based on the React Native framework and is therefore strongly influenced by web-based concepts (Viro Community, 2022). Both solutions are open source and enable cross-platform implementation.

Wikitude, Vuforia, and Kudan AR SDK are popular commercial solutions to implement cross-platform AR applications. They are not open source or free to use; therefore, a license needs to be acquired upon integrating them into an application. All three offer a Unity game engine plugin to enable cross-platform implementation. AR Foundation and artoolkitX follow a similar approach, but unlike the previously mentioned, both are freely usable.

Three frameworks were selected for detailed comparison based on the previously discussed requirements. The selected AR frameworks are AR Foundation, AR.js and ViroReact. All three enable cross-platform implementation and are also freely usable. In the case of AR.js and ViroReact, it is even possible to access the source code in order to better understand and, if necessary, adapt individual functions. The framework artoolkitX was also initially considered, but since the documentation and additional support are rather limited, the framework was not selected for the more detailed comparison. However, based on the previously defined requirements, it would also be suitable for the use in a CS project.

3.3 Prototype implementation

Prototypes were implemented for the three selected AR frameworks (AR Foundation, AR.js, ViroReact). In the process, different tracking techniques were examined to draw conclusions about which AR framework is suitable for specific use cases in (G)CS projects. The tracking techniques considered are two vision-based approaches (marker-based and image recognition) and one sensor-based approach (location-based).

The AR Foundation prototype was implemented by using the cross-platform game engine Unity. The AR Foundation package does not implement any AR functionalities itself, but rather provides an interface to use the native AR components on the different target platforms. This allows a straightforward implementation for Android and iOS, as only one application needs to be written, which, depending on the target platform, uses either ARCore or ARKit functionality in the background (Unity Technologies, 2022).

In contrast, AR.js is a purely web-based framework and is written entirely in JavaScript. An AR application implemented with the AR.js framework can therefore be executed directly in the browser, regardless of the target platform. The only requirements are that WebGL and WebRTC must be supported, which is already integrated in common mobile browsers. The AR.js framework can therefore be easily used to add AR functionality to an existing web-based application, since only two JavaScript libraries need to be loaded (AR.js Org, 2022).

ViroReact is based on React Native and can be compiled for both Android and iOS as target platforms. The framework can be used for creating AR and VR applications. Similar to AR Foundation, the native AR components ARCore and ARKit of the target platforms are used in the background for the implementation of AR functionality (Viro Community, 2022).

4. RESULTS & COMPARISON

The prototypes were tested on Android and iOS devices to confirm the cross-platform functionality. For this purpose, everyday smartphones were deliberately used rather than high-end devices, as it can be assumed that devices of varying quality will be used by participants of a CS project. On all devices, the settings have been adjusted so that the best possible location determination can be achieved. In total, three different devices were used to test the prototypes: A Samsung Galaxy A51 device with Android version 11, an older Samsung Galaxy S8 with Android version 9, and an iPhone SE (1st Generation) with iOS version 14.4 installed.

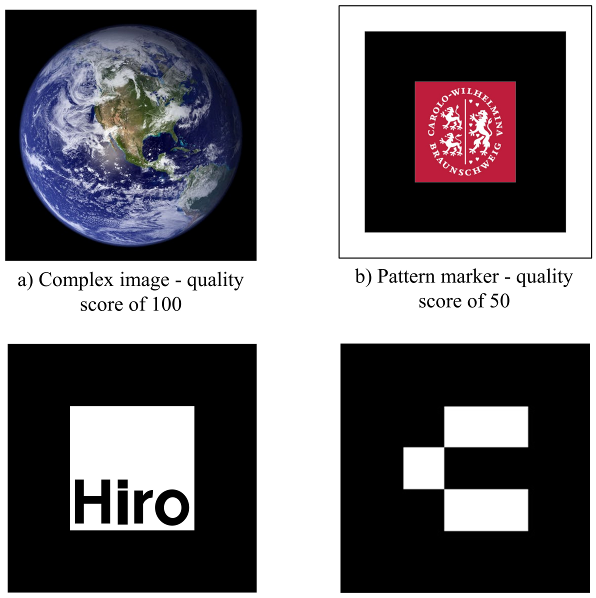
In the following, the results of the tests with the prototypes are presented. The implementation differences between the selected AR frameworks will be discussed, as well as the usability, speed, and accuracy of the resulting application. The results also indicate which AR Framework is best suited for each of the three tracking techniques considered.

4.1 Marker-based tracking

Marker-based tracking is widely used in AR applications. The user scans a marker with the mobile application and additional content is displayed in the AR scene. Since the markers are kept very simple, they can be detected particularly quickly. Due to its ease of use, the method is also suitable for participants who have no previous experience with AR. Marker-based tracking should therefore not be ignored in the context of CS, because it is the simplest and most straightforward way to integrate AR functionality into a project.

Marker-based tracking was implemented for all three of the selected frameworks and tested on the different devices. Markers of varying complexity (see Figure 1b-d) were used to display 3D objects in the AR scene. It is noticeable that the AR.js framework explicitly differentiates between marker-based tracking and image recognition during implementation. This is reflected in the need to integrate individual libraries into the prototypes for each of the two approaches. According to the documentation, it is not possible to use both approaches within one application. The implementation is therefore clearly different from the implementation in the ViroReact and AR Foundation frameworks. In these frameworks, the implementation of the two vision-based approaches follows the same principle, so there is no need to distinguish whether a marker or a more complex image is to be used for tracking.

The documentations of all frameworks emphasise that the quality of the markers (and images) plays a crucial role and determines how quickly and reliably the marker is recognised. For the tests three different marker types were used: Hiro marker, barcode marker and pattern marker. The Hiro marker is the default marker of the AR.js framework and can be seen in Figure 1c. Barcode markers are automatically-generated matrix-based markers. The dimensions of the matrix determine how many codes can be used to generate the markers. A simple example of a marker is shown in Figure 1d. It is a barcode marker with the dimension 3x3 and the code 6. The pattern marker, on the other hand, consists of a simple custom image surrounded by a black border (Figure 1b). Regardless of the marker type selected, all markers should always have a high contrast, and therefore black and white markers are always a good choice.



a) Complex image - quality score of 100

b) Pattern marker - quality score of 50

c) Hiro marker - quality score of 0

d) Barcode marker – quality score not calculable

Figure 1. ARCore quality score for selected reference images/markers

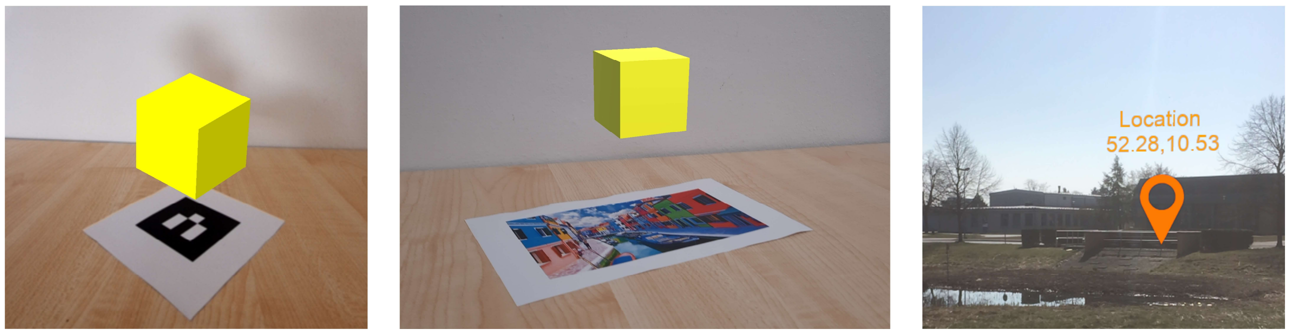
Figure 1 also shows the ARCore quality score for each marker/image. The score indicates how suitable the marker/image is for recognition with mobile devices that use ARCore in the background. The higher the score is, the better the recognition works. To achieve a high score, the image should have a wide range of geometric features and contain as few recurring patterns as possible. The score takes a value between 0 and 100, whereby ARCore recommends a value of at least 75 (Google Developers, 2022). The barcode marker is a negative example of a well-chosen marker, since it contains so few features that no ARCore quality score can be calculated. The more complex image in Figure 1a, on the other hand, is very suitable, which is reflected in a score of 100.

To find out which framework recognises the markers the fastest, a test of all frameworks was performed with the different mobile devices. The time between the moment when the marker is completely in the camera's view until the display of the 3D model was measured. Since the time was measured manually, the values do not reflect the exact time because the human reaction time is included. However, since this reaction time is included in all measured values, the numbers can certainly be compared to get a first impression of the speed of the individual frameworks. The first detection of the marker after opening the application usually takes longer than the subsequent ones, therefore the two cases were distinguished and measured individually. The marker was always moved to a different position in the scene before a subsequent detection. In order to make the test as realistic as possible, a test person with only limited prior experience in dealing with AR was chosen, since it is to be expected that some participants in CS projects also have never used AR on their own mobile device.

AR.js detects the markers quite fast and does not lose the tracked object even when moving the smartphone (see Table 2).

		Marker-based Tracking (average time until … detection)		**Image Recognition (average time until … detection)**	
Framework	**Mobile Device**	**first**	**subsequent**	**first**	**subsequent**
AR.js	Samsung Galaxy A51	2.21 s	2.01 s	7.25 s	3.11 s
	Samsung Galaxy S8	1.04 s	0.55 s	2.52 s	1.72 s
	iPhone SE	2.22 s	1.63 s	12.80 s	1.40 s
		Ø 1.82 s	**Ø 1.40 s**	**Ø 7.52 s**	**Ø 2.08 s**
AR Foundation	Samsung Galaxy A51	1.24 s	1.74 s	1.63 s	1.33 s
	Samsung Galaxy S8	1.01 s	0.90 s	0.85 s	0.84 s
	iPhone SE	1.25 s	2.78 s	1.55 s	1.63 s
		Ø 1.17 s	**Ø 1.81 s**	**Ø 1.34 s**	**Ø 1.27**
ViroReact	Samsung Galaxy A51	4.64 s	2.46 s	2.74 s	1.58 s
	Samsung Galaxy S8	2.76 s	2.45 s	2.66 s	1.89 s
	iPhone SE	3.23 s	1.76 s	1.09 s	1.30 s
		Ø 3.54 s	**Ø 2.22 s**	**Ø 2.16 s**	**Ø 1.59 s**

Table 2. Comparison of the average time (in seconds) required by the frameworks to detect a marker or image and display the corresponding 3D model



a) Marker-based tracking – implemented with AR.js

b) Image Recognition– implemented with AR Foundation

c) Location-based tracking – implemented with ViroReact

Figure 2. Examples of the results of the three different tracking techniques

The average time for all three devices is 1.82 seconds for the first detection and 1.4 seconds for all subsequent detection. A visual example of the marker-based approach with AR.js can be seen in Figure 1a. Overall, it is surprising that the newest device does not perform better than the older ones. The Samsung Galaxy S8 delivers the best results for all three frameworks and recognises the markers the fastest. The framework ViroReact performs the weakest and takes the longest to display the 3D objects in the scene. On average, it takes 3.54 seconds for the first detection and 2.22 seconds for all subsequent ones. The AR Foundation framework performs even better than AR.js. Here, the objects were rendered so quickly that it was sometimes difficult to measure the time meaningfully. With AR.js and ViroReact, a trend can be observed that all subsequent detections are significantly faster than the first one. This trend cannot be seen with the AR Foundation framework, but this may be because recognition is generally very fast. It should be noted that the simple barcode marker only worked in the web-based solution (AR.js) and on the iOS device, because the marker's ARCore quality score needs to be at least 0 to be detected with Android devices.

The results show that marker-based tracking can be a good fit for CS projects. Due to the fast and reliable recognition of the markers, participants can learn the process quickly, and this contributes to a good experience. However, since the lighting conditions are crucial for the recognition of the markers, the method should be used mainly indoors. For outdoor projects, black and white markers are preferable because they have good contrast and are better recognised than coloured markers.

4.2 Image recognition

Image recognition follows the same principle as marker-based tracking. The only difference is that more complex images are used to place the 3D objects into the scene. The advantage for CS projects is that real images can be used instead of artificial markers. For example, images already placed on an information board, in a leaflet or on a website can be utilised to display further information in the AR scene. The disadvantage is that it can be challenging for the participant to detect the image with their mobile device. Unlike marker-based tracking, it can be difficult to find the correct angle and distance to recognise the complex image. Furthermore, image recognition is more influenced by lighting conditions, which, as mentioned before, is strongly related to the colour representation.

Image recognition was implemented for all three frameworks. Two complex images were used to test the approach. One shows the earth and has an ARCore quality score of 100 (Figure 1a). The other one shows a canal surrounded by colourful buildings. The second image has an ARCore quality score of only 30 and can be seen in Figure 2b. Selecting a suitable image is not an easy task, so it is advisable to check the documentation beforehand to determine which image properties are important. The ARCore quality score can be used as a first reference value.

To test the approach, the same procedure as for marker-based tracking was used. The measured times until the recognition of the image and the display of the 3D model can be seen in Table 2. It is evident that the AR.js framework takes the longest to recognise the complex images. On average, it takes 7.52 seconds for the first image recognition and 2.08 seconds for all subsequent ones. Thus, the framework performs significantly slower than the other two. AR Foundation recognises the images the fastest. Occasionally, the image does not even have to be completely in the camera's view to be recognised. It is also noticeable that the AR Foundation framework works very consistently with all test devices. The Samsung Galaxy S8 is again slightly faster, but overall the experience is very good on all devices. ViroReact delivers similarly fast results as AR Foundation. Here, it is notable that the iPhone SE recognises the images much faster than the Android devices. Regardless of the framework and the device, it was observed that the image showing the earth was always recognised somewhat quicker than the second image. This confirms ARCore's recommendation to carefully select an image with a high quality score. Nevertheless, the second image was still reliably recognised by the AR Foundation and ViroReact frameworks despite the quality score of 30.

A comparison with marker-based tracking clearly shows that the AR Foundation and ViroReact frameworks recognise more complex markers/images much faster and reliably display the 3D object at the correct position even when the user moves the mobile device quickly. AR.js, on the other hand, works best with markers that are as simple as possible. The more complex the marker/image becomes, the more patience and experience the user needs to recognise it. The framework should therefore not be used for complex images in CS projects, so that participants do not abandon the AR experience due to frustration.

4.3 Location-based tracking

The location-based approach probably represents the greatest potential for GCS projects. Objects can be placed in the participant's environment based on geographic coordinates, allowing the project to cover a large area, in contrast to marker-/image-based approaches. The approach is thus suitable, for example, for guiding participants to a specific geographic location, for displaying data already collected at that position, or for giving participants further instructions once they have reached the position. A conceivable example could be an explanatory video that is shown to the participant in the AR environment, while he simultaneously sees the real environment in which the data is to be collected. The only prerequisite is that the participant has a mobile device with a GPS sensor and additional sensors to determine the device’s orientation.

In this study, simple 2D objects were placed in the AR scene based on geographic coordinates. An example of the result can be seen in Figure 2c. This simple approach gives a first indication whether the frameworks allow placement in relation to the geographic position of the participant. Generally, it can be noted that positioning by means of geographic coordinates can be implemented with all three frameworks. However, as expected, the accuracy of the positioning depends heavily on the sensors in the used mobile devices. It should therefore be carefully examined by each GCS project whether the use of a location-based tracking approaches is feasible and leads to the desired results.

The AR.js framework directly integrates the functionality to place objects based on geographic coordinates. The documentation describes how the implementation can be carried out. It should be noted that the documentation is very limited. In addition, the libraries mentioned do not match those in the referenced examples, which can cause confusion and complicates the implementation. However, since AR.js is an open source framework, it is of course always possible to check the source code to clarify any ambiguities. Of the three AR frameworks to be compared, AR.js is the only one that offers a location-based approach out-of-the-box. It is therefore the easiest solution to integrate a location-based approach into GCS projects.

AR Foundation and ViroReact do not support this functionality without further custom implementation. For both frameworks, a script must be written that first accesses the current position of the mobile device and then converts the geographic position of the device and the object into a Cartesian reference system. Only then can the objects be placed in the scene by calculating the distance between the two points. Afterwards, the position of the object in the scene should be adjusted using the compass heading to account for the orientation of the mobile device.

The prototype tests reveal that the location-based tracking approach is error-prone for all three compared frameworks. The distance of the placed objects in relation to the mobile device depends on the accuracy of the geographical coordinates determined by the GPS sensor. If the position is inaccurate, then the objects will be placed too close or too far away. Therefore, in order to increase the accuracy of the placement, the positioning of the objects should be delayed until the values of the GPS sensor have reached an acceptable accuracy level. The acceptable level of accuracy depends heavily on the use case and should be determined on an individual basis. If the AR functionality is used to show a participant the position at which new data should be collected, then an inaccuracy of a few meters is of course more significant than if data that has already been collected is to be visualised for other participants at its approximate measurement location. During the practical tests with the prototypes, an accuracy of 5-10 meters was observed. Lower values are of course preferable, but they lead to a longer waiting time for the participant until the objects are placed in the AR scene.

The orientation and placement of the objects in relation to the participant holding the mobile device is an even greater challenge. In order for the objects to be displayed in the correct position, it is not sufficient to only know the distance between the mobile device and the geographical position. Rotation must also be considered for the object to be placed on the correct side of the participant. For this purpose, the position must be corrected using the compass heading. The tests demonstrated that the compass measurements of the three test devices are rather inaccurate. By calibrating the compass beforehand, this problem can be corrected to some extent, but the determination of the orientation still remains imprecise and erratic. This results in objects being placed at approximately the correct distance from the mobile device, but not with the correct angle compared to the orientation of the device. For the frameworks that use ARKit in the background, the orientation does not have to be determined separately when the application is used on iOS devices. ARKit already integrates a setting that allows the negative z-axis of the AR scene's coordinate system to automatically align to the north. When testing the ViroReact framework with the iPhone, it was observed that the placement of the objects improved significantly when the above-mentioned setting was applied and the orientation was not calculated via the compass heading. Unfortunately, a comparable setting currently does not exist for ARCore.

5. CONCLUSION

In conclusion, it can be observed that all three compared AR frameworks have potential for integration in (G)CS projects, but the choice of framework depends strongly on the functionality to be implemented and the already existing technical infrastructure. For projects that already offer a web-based solution and plan to add simple marker-based AR functionality, the AR.js framework is the right choice. The customisation effort would be minimal and the AR functionality could add significant value for participants. If more complex images are to be used for tracking, then AR Foundation or ViroReact are a better fit as the recognition of the images is much more reliable. Without further customisation location-based tracking should only be used for coarse positioning, for example, to show participants in a GCS project the approximate position they should move towards. To use the method for more targeted purposes, the custom scripts would need to be extended and refined.

Of course, the comparison presented has also its limitations. On the one hand, the non-functional requirements should be extended to enable a more precise selection of AR frameworks. So far, only two basic requirements for the use of frameworks in CS projects have been identified, which could lead to a lack of consideration of other suitable AR frameworks. Of course, the final choice of an AR framework strongly depends on the individual requirements of a CS project. The comparison presented here should only serve as an orientation and simplify the selection process. On the other hand, the presented functional approaches should be further deepened and tested. The AR frameworks offer a wide range of possible applications and the tracking methods compared represent only a fraction of them. Therefore, these should be tested and analysed in more detail to make recommendations as to which AR frameworks are suitable for more individual functional requirements. Furthermore, hardware aspects of the individual AR frameworks should be evaluated, since, for example, power consumption plays a significant role in AR applications. Especially when applying the location-based approach, it becomes clear that the constant request of the geographic position has a strong impact on the power usage. This should be examined and deepened in further studies.

The comparison of the AR frameworks and the subsequent tests with the prototypes have also shown that the geographical position, which is determined via the GPS receiver of everyday mobile devices, might be too inaccurate for specific applications. It remains to be seen whether the positioning accuracy of everyday smartphones will improve in the future. The announcement of a meter-level accuracy for Android smartphones, realised by the partnership of Qualcomm and Trimble (Qualcomm Technologies, Inc., 2022), is a step in the right direction and can thus also have a positive effect on the use of AR applications in GCS projects. Nevertheless, any GCS project should thoroughly test whether the current accuracy of the positioning is sufficient before integrating a location-based AR component into the project. Including an AR component in a (G)CS project should always have a clear benefit for the participants and the project itself. The technology should only be used if it advances the project.

REFERENCES

Albers, B., De Lange, N., Xu, S., 2017: Augmented citizen science for environmental monitoring and education. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci*, XLII-2/W7. doi.org/10.5194/isprs-archives-XLII-2-W7-1-2017.

Amin, D., Govilkar, S., 2015: Comparative study of augmented reality SDKs. *International Journal on Computational Science & Applications*, 5(1), 11-26.

AR.js Org, 2022. Ar.js – Augmented Reality on the web. ar-js-org.github.io/AR.js-Docs/ (23 March 2022).

Burkard, S., Fuchs-Kittowski, F., Himberger, S., Fischer, F., Pfennigschmidt, S., 2017: Mobile Location-Based Augmented Reality Framework. *12th International Symposium on Environmental Software Systems (ISESS)*, doi.org/10.1007/978-3-319-89935-0_39.

Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., Ivkovic, M., 2011: Augmented reality technologies, systems and applications. *Multimedia tools and applications,* 51, 341–377. doi.org/10.1007/s11042-010-0660-6.

Chatzopoulos, D., Bermejo, C., Huang, Z., Hui, P., 2017: Mobile augmented reality survey: From where we are to where we go. *Ieee Access*, 5, 6917–6950. doi.org/10.1109/ACCESS.2017.2698164

Google Developers, 2022. ARCore – Add dimension to images. developers.google.com/ar/develop/augmented-images (25. March 2022).

Hasler, O., Blaser, S., Nebiker, S., 2020: Performance Evaluation of a Mobile Mapping Application Using Smartphones and Augmented Reality Frameworks. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2, 741-747. dx.doi.org/10.5194/isprs-annals-V-2-2020-741-2020

Herpich, F., Guarese, R. L. M., Tarouco, L. M. R., 2017: Comparative Analysis of Augmented Reality Frameworks Aimed at the Development of Educational Applications. *Creative Education*, 8, 1433-1451. dx.doi.org/10.4236/ce.2017.89101.

Milgram, P., Kishino, F., 1994: A taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems,* 77(12), 1321–1329.

Oufqir, Z., El Abderrahmani, A., Satori, K., 2020: ARKit and ARCore in serve to augmented reality. *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. doi.org/10.1109/ISCV49265.2020.9204243.

Qualcomm Technologies, Inc., 2022. Qualcomm and Trimble Introduce Meter-Level Location Accuracy for Smartphones. www.qualcomm.com/news/releases/2022/03/22/qualcomm-and-trimble-introduce-meter-level-location-accuracy-smartphones (28 March 2022).

Rabbi, I., Ullah, S., 2013: A survey on augmented reality challenges and tracking. *ACTA GRAPHICA Journal for Printing Science and Graphic Communications*, 24 (1-2), 29-46.

Sansom, E., Ridgewell, J., Bland, P., Paxman, J., 2016: Meteor reporting made easy – The Fireballs in the Sky smartphone app. *Proceedings of the International Meteor Conference (IMC)*, 267-270.

Sermet, M.Y., Demir, I., Kucuksari, S., 2018: Overhead power line sag monitoring through augmented reality. *2018 North American Power Symposium (NAPS)*. doi.org/10.1109/NAPS.2018.8600565.

Unity Technologies, 2022. AR Foundation. unity.com/unity/features/arfoundation (25 March 2022).

Viro Community, 2022. ViroReact - Overview. viro-community.readme.io/docs (23 March 2022).

Wiggins, A., 2013: Free as in puppies: compensating for ICT constraints in citizen science. *Proceedings of the 2013 conference on Computer supported cooperative work*, 1469-1480. dx.doi.org/10.1145/2441776.2441942.

APPENDIX

The source code of the prototypes is available in the following GitLab repository and can be cloned and adapted for own projects: https://git.rz.tu-bs.de/users/cosima.berger/projects.