Cloud-Based Cross-Platform Collaborative AR in Flutter

Lars Carius* Christian Eichhorn† David A. Plecher‡ Gudrun Klinker§

Technical University of Munich

Abstract

Augmented Reality (AR) has progressed tremendously over the past years, enabling the creation of collaborative experiences and real-time environment tracking on smartphones. The strong tendency towards game engine-based approaches, however, has made it difficult for many businesses to utilize the potential of the technology. We present a novel collaborative AR framework aimed at lowering the entry barriers and operating expenses of AR applications. Our framework includes a cross-platform and cloud-based Flutter plugin combined with a web-based content management system allowing non-technical staff to take over operational tasks such as providing 3D models or moderating community annotations. To provide a state-of-the-art feature set, the AR Flutter plugin builds upon ARCore on Android and ARKit on iOS and unifies the two frameworks using an abstraction layer written in Dart. We show that the cross-platform AR Flutter plugin performs on the same level as native AR frameworks in terms of both application-level metrics like CPU and RAM consumption and tracking-level qualities such as keyframes per second used by the SLAM algorithm, detected feature points, and area of tracked planes. Our contribution closes a gap in today's technological landscape by providing an AR framework seamlessly integrating with the familiar development process of cross-platform apps. Using the AR Flutter plugin and the accompanying content management system, AR can be used as a tool to achieve business objectives and is not restrained to stand-alone single-purpose apps anymore, triggering a potential paradigm shift for previously complex-to-realize applications of AR, e.g. in production and planning. The AR Flutter plugin is fully open-source, the code can be found at: `https://github.com/CariusLars/ar_flutter_plugin`.

Index Terms: Augmented Reality, Collaboration, Cross-Platform, Cloud, Flutter, Open-Source——
—

1 Approach

Augmented Reality (AR) allows for a fundamentally new means of communication: Anchoring digital information in the real world, retrievable by anyone visiting the respective location. On-demand 3D scans using the smartphone camera overcome the need for hand-built 3D environment models and make this technology widely applicable. However, heavy reliance on game engine-based development toolchains like Unity has left behind developers of business applications and has created a technological landscape dominated by the entertainment industry. We discovered that the technology can create considerable value once it is used as a tool in operative business and not primarily as a means of entertainment. Marking drop-off locations for deliveries to chaotic construction sites using AR is one of many examples that proved to be a profitable business case.

*e-mail: lars.carius@tum.de
†e-mail: christian.eichhorn@tum.de
‡e-mail: plecher@in.tum.de
§e-mail: klinker@in.tum.de

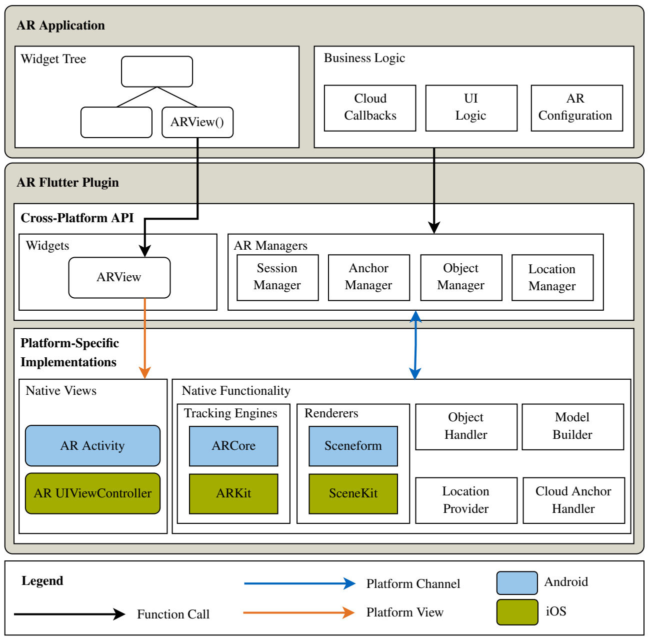

Figure 1: High-level software architecture of the AR Flutter plugin

To utilize this potential, AR functionality needs to be embedded - as just one of many features - into applications that serve a larger purpose. With the frameworks currently available, this is not sufficiently possible. There are a plethora of approaches trying to solve this challenge, but they either lack the inclusion of cutting-edge features such as anchoring objects to real-world places or heavily restrict generality and thus applicability by focusing development on game scenarios only.

We present a cross-platform, open-source collaborative AR framework for Flutter to enable more businesses and app developers to utilize the potential of location-based AR without having to renounce technical possibilities and performance advantages that originate from using an established cross-platform app development toolchain. With our work, we aim to facilitate the development of complex AR-based workflows such as facility design and management in production and planning scenarios while, at the same time, alleviating the need for excessive computing power and pre-defining the territory of deployment.

2 Implementation

The AR Flutter plugin's architecture is composed of two parts (Fig. 1): A unified, cross-platform API providing an interface to applications using the plugin, and platform-specific implementations for Android (Kotlin) and iOS (Swift) built upon ARCore and ARKit to ensure continuous access to cutting-edge functionality.
The exposed section of the framework contains widgets that can be included in a client app's widget tree to enhance the UI and AR managers that handle all functionality and logic related to AR

and serve as the control instruments of the plugin. The session manager handles tracking configurations, debugging options like plane visualization, and callbacks for hit-testing and gesture events. The object manager handles the plugin's ARNodes which abstract platform-specific functionality and allows to add, modify, or remove nodes based on GLTF2- or GLB-format 3D models that can be compiled into the app bundle or loaded during runtime from the file system or the internet. We salvaged functionality from existing single-platform Flutter AR plugins [1] to build upon existing research. To enable collaborative AR experiences [2], the anchor manager contains functionality to upload and download anchor objects from a cloud service using the Google Cloud Anchor API. Local 3D scans used to define anchors can be stored in a cloud system and the current scene can be compared to already uploaded anchors in order to download content previously placed in the scene. To enable efficient location-based anchor querying, the location manager provides GPS coordinates of the device. The plugin's node and anchor object can be serialized and synced with a database.

The largely cloud-agnostic architecture of the AR Flutter plugin enables the use of external content management systems to reduce the running costs of commercial AR applications. We demonstrate a web-based system allowing the modification of both content used in the app's UI, such as 3D model files, and data uploaded by users, such as AR annotations and corresponding text, to be managed remotely. We utilized the open-source software Firetable[2]. New 3D models can easily be added to existing applications in the spreadsheet-like interface. This content management architecture enables the operations team to modify content without having to task the technical development team with updates to the source code.

3 Evaluation

The usage of cross-platform app development frameworks has many benefits, however, it can require additional resources to achieve performance comparable to native applications due to additional software layers being introduced.
To assess whether or not AR app performance is significantly impacted by the use of Flutter and the AR Flutter plugin, we benchmarked our framework against comparable native approaches.

Our results indicate that the usage of Flutter and the AR Flutter plugin incurs overhead in both build times and application sizes. On Android, both measurements roughly double when using the cross-platform approach. On iOS, the build time increases even more drastically, while application sizes also double.
To assess performance, we tracked average and maximum CPU and RAM utilization and, on iOS, average frame time. We observed that on Android, the cross-platform application based on the AR Flutter plugin performs on the same level as the native Android application. On iOS, solely the average and maximum RAM utilization exhibit an increase of roughly 100 MB for the Flutter application.
To reduce the processing load on the device and avoid jerking, AR frameworks can analyze incoming sensor readings less extensively or even skip them entirely, resulting in worse augmentation quality while measurements like framerate remain unchanged.
Thus, to test the quality of the augmentation and, therefore, the scene understanding performance offered by the AR Flutter plugin, we conducted a series of experiments directly measuring the number of features processed by the benchmarking applications, the total number of AR frames (quantity of SLAM updates), the number of detected feature points per AR frame (quality of SLAM updates), and the total detected plane area in a fixed setting.

[1]G. M. D. Francesco. ArCore Flutter Plugin; O. Leuschenko. ARKit Flutter Plugin
[2]FiretableProject. firetable, 2021.

	Total Number of AR Frames	**Total Number of Feature Points**	**Feature Points per AR Frame**	**Total Tracked AR Plane Area**
Android Native AR Application	298 ± 12	41000 ± 3107	137 ± 8	$8.19 \pm 0.67\ m^2$
Flutter AR Application	300 ± 9	38057 ± 2324	126 ± 7	$8.15 \pm 0.89\ m^2$

Table 1: Tracking quality benchmark on Android

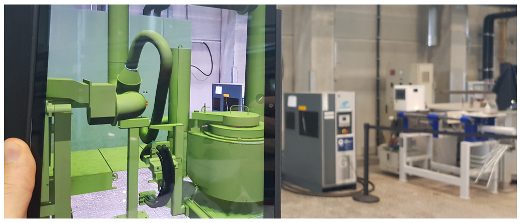
Figure 2: Augmentation of an industrial site

Table 1 shows mean and standard deviation of 5 consecutive runs of the benchmark on an Android device. The evaluation of low-level tracking features to measure the plugin's impact on a SLAM performance level yields a similar number of AR frames for both applications. While the total number of feature points, and, consequently, the average number of feature points per AR frame, is slightly lower in the cross-platform application, the values' mean $\pm$ standard deviation intervals still overlap. The plane area detected by the tracking algorithms directly translates to the area available for the user to place AR objects onto, rendering it a valuable criterion for AR performance. The measurements are almost equal for our and the native approach. Based on this data, it can be concluded that the SLAM algorithm's performance suffered a negligible impact through the additional layer of abstraction our plugin introduces, allowing for the same level of augmentation quality in our framework as in state-of-the-art native apps.

4 Impact

We introduced a novel cross-platform, cloud-based collaborative AR framework that allows non-expert developers to utilize the potential of AR as a feature of their application. Our analyses find no significant performance difference between a native approach and our framework. We already proved our plugin's versatility in an industrial machinery augmentation project (Fig. 2).

Our contribution facilitates a paradigm shift in the way complex location-based AR workflows can be established: Instead of having to resort to preparation-heavy approaches involving 3D models of the deployment environment and a computation backbone (see [1]), our plugin allows to utilize AR annotations for tasks like facility management in unknown terrain on standard smartphone hardware. On a higher level, our development can be viewed as an effort to integrate the multifaceted realm of AR frameworks, many of which are controlled and restricted by large software companies, into a common, open-source, and cross-platform API. Our approach builds on a modular architecture, which, combined with a reliance on efficient patterns and the support of a large community of developers, has the potential to bundle the technological capabilities of various frameworks into a powerful single source of AR functionality.

References

[1] F. Baek, I. Ha, and H. Kim. Augmented reality system for facility management using image-based indoor localization. *Automation in Construction*, 99:18–26, 2019. doi: 10.1016/j.autcon.2018.11.034

[2] H. Kaufmann. Collaborative Augmented Reality in Education. Technical report, 2003.