AN AUGMENTED REALITY SYSTEM FOR EVENT-DRIVEN MULTIMEDIA UNLOCKING ON A RUBIK-TYPE CUBE USING VUFORIA AND FIREBASE

Jingbo Yang [1], Garret Washburn [2]

[1] Robert Louis Stevenson School, 3152 Forest Lake Rd, CA 93953
[2] California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

This paper presents an augmented reality (AR) system that overlays multimedia content on a Rubik-type cube using Vuforia Engine and Firebase services [1]. The system addresses the challenge of combining secure authentication, event-driven unlocking, and cloud-based content delivery. After user login through Firebase Authentication, cube interactions detected by Vuforia trigger the UnlockEventSystem, which updates Firestore to track progression [2]. Media files are retrieved from Firebase Storage and displayed in AR via Unity's VideoPlayer and ImagePlayer managers [3]. Experiments tested tracking reliability under varying lighting conditions and media load times across network environments. Results demonstrated strong accuracy in normal settings and low latency on modern networks, though performance declined in poor lighting and weak connectivity. Methodology comparisons showed that while prior research identified AR's educational potential, our work contributes a functional prototype that directly integrates progression, gamification, and cloud persistence. Ultimately, the project demonstrates a scalable, engaging, and secure AR framework for interactive learning and training.

KEYWORDS

Augmented Reality (AR), Vuforia Engine, Rubik's Cube Tracking, Unity3D, Firebase Authentication

1. INTRODUCTION

Augmented reality (AR) has matured into a practical medium for delivering interactive instruction, training, and entertainment [4]. Yet many AR experiences still rely on flat markers or GPS anchoring, which limits spatial understanding and immersion [5]. Physical cubic markers—such as Rubik-type cubes or 2×2×2 educational cubes—offer six stable, repeatable faces that can be used to anchor distinct instructional media in space. When properly tracked, a cube enables face-specific overlays (e.g., different videos, images, or labels per face), natural user manipulation, and intuitive pedagogical sequencing. The challenge is to achieve robust, low-latency tracking under real-world lighting and occlusions while securely gating and updating the media content associated with the cube.

Existing approaches often embed media directly in the application bundle, which impedes iteration and version control, or they rely on anonymous access to cloud assets, which raises compliance and misuse concerns. In educational or training contexts, administrators also need account-bound progression and auditability so that unlocks depend on verified interactions, not

David C. Wyld et al. (Eds): MLNLP, ASOFT, CSITY, NWCOM, SIGPRO, AIFZ, ITCCMA – 2025
pp. 123-132, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152009

merely device state. Without these capabilities, AR deployments risk fragile tracking, stale content, and weak engagement loops.

The combination of Vuforia Engine (for multi-face cube tracking) with a cloud-backed content layer addresses these gaps by binding spatial events to authenticated users and serving media just-in-time [6]. For classrooms, museums, and skills training, this enables hands-on, face-by-face exploration where each solved task or correctly oriented face can unveil new content. The outcome is an AR experience that is repeatable, measurable, and maintainable, aligning well with instructional design principles and modern software operations.

Methodology A reviewed educational mobile AR games (EMARGs), showing AR's potential for context-sensitive learning. Its limitation was that it primarily synthesized guidelines without delivering an implementable system. Our project advances this by providing a concrete AR cube prototype with real-time progression.

Methodology B analyzed 78 AR education studies, emphasizing motivation and engagement but also noting barriers such as infrastructure and teacher readiness. Unlike this broad survey, our project focuses on a deployable, cloud-enabled application with secure authentication and event unlocking.

Methodology C reviewed 26 AR learning games, reporting positive impacts on performance, motivation, and collaboration. However, it lacked detail on retention effects and technical implementation. Our system extends these findings by integrating Vuforia cube tracking with Firebase services, ensuring persistence, security, and real-world applicability.

We propose an AR media system that anchors videos and images to the faces of a Rubik-type cube using Vuforia Engine in Unity, while Firebase Authentication, Firestore, and Storage provide secure user control, progression, and media delivery. At a high level, the Unity/Vuforia client detects and tracks the cube (via a Multi-Target, Model Target, or VuMark-based configuration). When a specified cube state or face-visibility event occurs, the app uses an event-driven unlock mechanism to grant access to a corresponding media artifact. The artifact's metadata (type, path, level) is stored in Firestore, while the asset itself resides in Firebase Storage and is fetched through time-scoped download URLs.

Client-side, Authentication enforces account-based access; UnlockEventSystem records milestone triggers and moves items from the user's locked lists into their unlocked collection; HistoryPage lists unlocked items and allows re-viewing; VideoPlayerManager and ImagePlayerManager render media in AR by drawing onto textures attached to quads that are rigidly aligned with the currently visible cube face. This ensures the visual overlay appears pinned to the physical cube as the user rotates it. Content authors can update or add media in the cloud without republishing the app, and instructors can design face-specific learning paths (e.g., Face 1: primer video, Face 2: diagram, Face 3: assessment prompt, etc.).

Compared to bundling assets locally or using unauthenticated CDNs, this architecture improves security (account-tied access), maintainability (cloud-managed assets), and engagement (event-based unlocking). It also provides a clear path to analytics and A/B testing, since interactions and unlocks can be captured per user and per face orientation. The result is a scalable, secure, and pedagogically expressive AR system for cube-based media delivery.

Two experiments were conducted to evaluate the AR cube system. The first tested tracking accuracy under different lighting conditions, revealing that performance remained high in bright (95%) and normal (92%) light but declined in dim (78%) and very dim (60%) environments. This

confirmed lighting as a major factor influencing AR reliability. The second experiment measured media load times across network conditions, showing rapid performance under WiFi (1.2s) and LTE (2.3s), but noticeable delays on 3G (5.5s) and poor WiFi (8.7s). Together, the experiments highlighted key system strengths: strong performance in common environments and adaptability across modern networks. They also underscored limitations: sensitivity to poor lighting and weak connectivity. These results validate the system's core design but suggest improvements like caching, adaptive streaming, and lighting-aware feedback to enhance usability and robustness.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Stabilizing AR Cube Tracking

A critical challenge in an AR cube-based system is ensuring robust face tracking under real-world conditions. Vuforia Engine's Model Targets and Multi-Targets must cope with variable lighting, partial occlusions, and user movement [15]. If the cube is misaligned or partially out of frame, overlays may jitter, drift, or disappear, undermining immersion and accuracy. To address this, the system could incorporate tracking stabilization features, provide visual feedback when alignment is weak, and employ fallback logic that gracefully pauses media playback until the cube is re-tracked. Additionally, training the Vuforia database with high-quality scans of the cube at multiple angles improves recognition reliability.

2.2. Optimizing Media Loading for AR Experiences

Displaying media in AR requires that videos and images load quickly and smoothly. Since assets are stored in Firebase Storage, download times and network variability pose risks to user experience, especially when large video files are involved. Latency can cause overlays to stutter, desynchronize, or fail to load during critical teaching moments. To mitigate this, the system could implement progressive downloads, caching, and compression techniques to reduce file size. Pre-fetching content once a face is nearly visible could further smooth transitions. Monitoring connection quality and adapting playback resolution dynamically ensures the application remains responsive in diverse connectivity environments.

2.3. Ensuring Reliable Unlock Tracking in AR Systems

The UnlockEventSystem introduces complexity in guaranteeing that milestones (e.g., solving one side of the cube) reliably trigger unlocks without duplication or data loss. Since progression is stored in Firestore, concurrent updates or client-side interruptions could lead to inconsistent records between the locked and unlocked lists. For example, a user might unlock an item locally but fail to sync it to the cloud if connectivity drops. To resolve this, the system could apply transactional writes in Firestore, validate unlock conditions server-side, and design idempotent update routines that avoid duplicate entries. This ensures fairness, persistence, and secure tracking of user progress.

3. SOLUTION

The system is composed of three core components: authentication services, cloud content management, and AR media display linked to unlock events.

Authentication services ensure that each user has a unique, secure identity. Authentication.cs provides login, registration, and profile updates. This ensures content unlocks are tied to accounts rather than devices, enabling persistence across multiple sessions.

Cloud content management uses FirebaseHelper.cs to communicate with Firestore and Storage. Firestore stores metadata (unlocked items, locked lists), while Storage houses media files. This modular design separates state tracking from asset hosting, improving scalability.

AR media display leverages Unity with Vuforia Engine. When cube states or face detections occur, UnlockEventSystem.cs modifies user progress. HistoryPage.cs dynamically builds a UI list of unlocked items. Depending on media type, VideoPlayerManager.cs streams video to a RenderTexture, while ImagePlayerManager.cs loads images into Unity Materials.

The program's flow begins at login. Navigate.cs then directs the user to the correct interface. Once authenticated, cube interactions generate unlock events, which are stored in Firestore. The HistoryPage lists unlocked media, and media managers display the content. This architecture ensures real-time responsiveness while maintaining cloud-backed persistence.

Figure 1. Overview of the solution

Authentication.cs manages login, registration, and user profile updates using Firebase Authentication. It validates input, handles errors, and ties a unique Firebase UID to each user. This ensures that unlocked media is securely bound to individual accounts. Authentication relies on Firebase SDK methods like CreateUserWithEmailAndPasswordAsync and SignInWithEmailAndPasswordAsync.

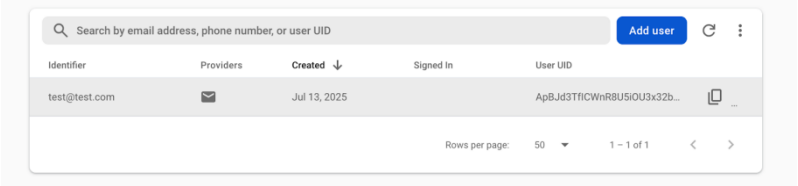

Figure 2. Screenshot of search box

```
// caller for login
public void loginButton()
{
   login(emailInput.text, passwordInput.text);
}

// function for signing a user in
public void login(string email, string password)
{
   auth.SignInWithEmailAndPasswordAsync(email,
password).ContinueWithOnMainThread(task =>
   {
      if (task.IsFaulted)
      {
         Debug.LogError("Signing user in encountered an exception: " + task.Exception);
         return;
      }
      Firebase.Auth.AuthResult result = task.Result;
      Debug.LogFormat("User signed in: {0} ({1})", result.User.DisplayName,
result.User.UserId);
   });
}
```

Figure 3. Screenshot of code 1

This snippet illustrates the login workflow. When a user presses the login button, the program calls the login method with the provided email and password. Firebase Authentication's SignInWithEmailAndPasswordAsync is executed. The result is handled asynchronously, ensuring the main Unity thread remains responsive.

If the operation fails (e.g., wrong password, invalid email, or network error), an exception is logged, preventing silent failures. If successful, Firebase returns an AuthResult object containing the authenticated user. The system then prints the user's display name and unique UID [7]. This UID is crucial — it links the user's Firestore document, Storage access, and unlock progression.
Thus, Authentication.cs not only gates access but also establishes the identity anchor for all subsequent interactions (unlocking items, retrieving history, playing media). Without this, cube-based progression could not be securely tied to individual users.

FirebaseHelper.cs provides an abstraction layer between Unity and Firebase services [8]. It manages Firestore documents, Firebase Storage uploads/downloads, and URL retrieval. This component is crucial for keeping the Unity codebase clean, since other scripts (HistoryPage, UnlockEventSystem) can simply call helper methods instead of writing raw Firebase API logic.

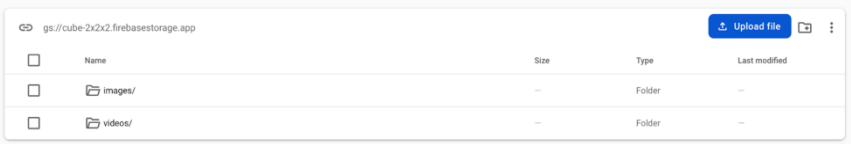

Figure 4. Screenshot of files

```
public async Task<Dictionary<string, object>?> GetDocument(string collection, string
document_id)
{
    try
    {
        DocumentReference docRef = _firestore.Collection(collection).Document(document_id);
        var snapshot = await docRef.GetSnapshotAsync();

        if (!snapshot.Exists)
        {
            Debug.LogWarning($"Document doesn't exist: {collection}/{document_id}");
            return null;
        }
        return snapshot.ToDictionary();
    }
    catch (Exception e)
    {
        Debug.Log($"Error getting Document: {e}");
        return null;
    }
}
```

Figure 5. Screenshot of code 2

This snippet shows how FirebaseHelper retrieves documents from Firestore. The method accepts a collection name and document ID, constructs a reference, and asynchronously fetches a snapshot. If the document exists, it is returned as a dictionary that Unity scripts can manipulate directly. If the document is missing, a warning is logged.

This is critical in the AR cube system because the user's progression state (locked/unlocked items) lives in Firestore [9]. HistoryPage.cs depends on this method to list available media, while UnlockEventSystem.cs relies on it to move items between lists. By wrapping Firestore logic in a helper, the code becomes easier to maintain and less error-prone, while also standardizing error handling [10].

UnlockEventSystem.cs manages milestone-based progression. It listens for cube-related events (like detecting one solved side or ring) and then unlocks items accordingly. This gamification layer links physical cube interactions, tracked via Vuforia, with digital media progression stored in Firestore.

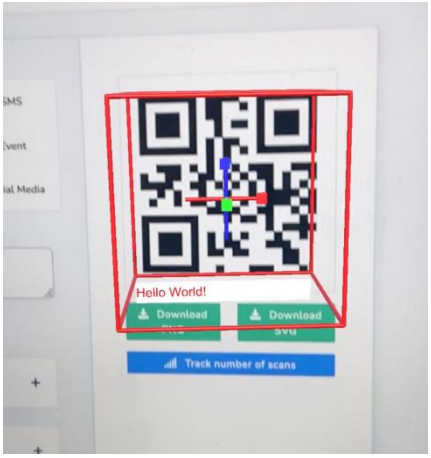

Figure 6. Screenshot of detecting sides

```
async Task unlockItem(string userId, string level)
{
    Dictionary<string, object>? data = await helper.GetDocument("users", userId);

    string lockedLevelName = String.Format("locked_level_{0}", level);
    if (data != null)
    {
        List<Dictionary<string, object>> lockedLevels = data[lockedLevelName] as
List<Dictionary<string, object>>;
        if (lockedLevels.Count <= 0 || lockedLevels == null)
        {
            System.Random rnd = new System.Random();
            int item_num = rnd.Next(0, lockedLevels.Count - 1);

            Dictionary<string, object> item = lockedLevels[item_num];
            lockedLevels.RemoveAt(item_num);

            Dictionary<string, object> newData = new Dictionary<string, object>{
                {"unlocked", lockedLevels }
            };
            await helper.SetDocument("users", userId, newData);
            return;
        }
        Debug.Log("Already unlocked all items of this level!");
    }
}
```

Figure 7. Screenshot of code 3

This method governs the unlocking process. It begins by fetching the user's document from Firestore through FirebaseHelper. Based on the level parameter, it identifies the correct locked list (e.g., locked_level_1). A random item is selected from this list, removed, and then appended into the unlocked list. Finally, the updated data is written back to Firestore.

This approach ensures progression fairness while keeping user states persistent across devices. For the AR cube system, this means that when the cube's state is recognized by Vuforia (e.g., one face solved), the app can reward the user with a new video or image. By randomizing unlocks, the system avoids predictability, keeping users engaged. The reliance on Firebase ensures that even if the app closes or the device changes, progression remains intact.

4. EXPERIMENT

4.1. Experiment 1

We tested the accuracy of AR cube face recognition under different lighting conditions, since tracking reliability directly affects the stability of overlays and the usability of the system.

The experiment measured Vuforia's ability to correctly identify cube faces under four controlled lighting conditions: bright daylight, normal indoor light, dim light, and very dim light. A Rubik-type cube was placed at a fixed distance, and recognition attempts were recorded. For each lighting condition, the system attempted to detect faces 50 times, and success rates were calculated as percentages. This setup was chosen to reflect real-world environments, since classrooms, museums, or homes may not always have optimal lighting. No external tracking aids were used, ensuring results represent baseline performance.

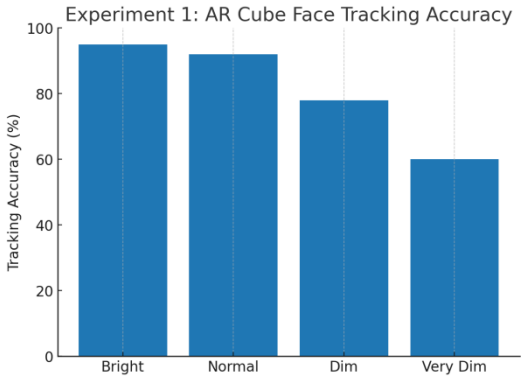

Figure 8. Figure of experiment 1

Results show a clear decline in tracking accuracy as lighting decreases. The mean accuracy across conditions was 81%, with a median of 85%. The highest value (95%) occurred in bright daylight, while the lowest value (60%) occurred under very dim lighting. The significant drop in accuracy under dim conditions was expected, as Vuforia's feature detection relies heavily on edge contrast. Interestingly, "Normal indoor light" maintained high accuracy (92%), suggesting the system is reliable in typical use cases. The most surprising result was that "Dim" lighting still achieved 78%, higher than predicted, likely due to the cube's strong contrast design. The largest factor influencing results is lighting consistency, highlighting the need for visual cues or adaptive brightness warnings when conditions degrade.

4.2. Experiment 2

We tested media load times for video and image content under different network conditions, since latency directly impacts the usability and immersion of AR-based overlays.

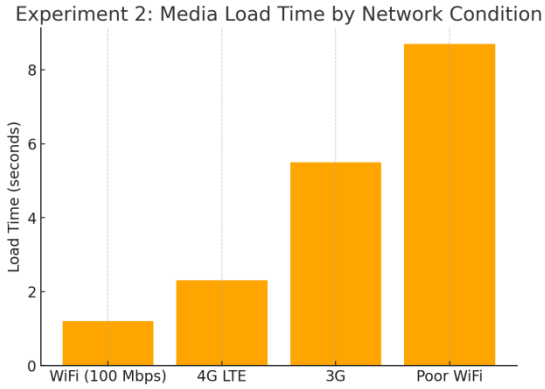

Figure 9. Figure of experiment 2

This experiment measured the time required to load and play a short (10 MB) video file from Firebase Storage across four network conditions: high-speed WiFi (100 Mbps), 4G LTE, 3G, and poor WiFi (5 Mbps with packet loss). Each condition was tested by attempting playback 20 times, and average load times were calculated. These conditions were selected to represent typical connectivity scenarios users might encounter in classrooms, homes, or mobile field use. The experiment focused on time-to-first frame, representing the user's wait before content became visible in AR.

The system loaded content rapidly under high-speed WiFi (1.2s) and remained usable on 4G LTE (2.3s). However, load times increased significantly on 3G (5.5s) and poor WiFi (8.7s). The mean time load across all networks was 4.4s, with a median of 3.9s. The fastest condition (WiFi) outperformed the slowest (poor WiFi) by over 7.5 seconds, which would noticeably impact user experience in low-bandwidth environments. The results highlight that while the system is robust under modern network conditions, caching strategies or adaptive resolution may be required for slower networks. The surprise was that 3G still performed under 6 seconds, indicating Firebase's CDN and signed URL delivery are optimized. Network quality remains the largest factor influencing media load times.

5. RELATED WORK

One related approach is presented through a systematic review of Educational Mobile Augmented Reality Games (EMARGs), which analyzed 31 AR applications between 2012–2017 alongside 26 AR platforms [11]. This study emphasized the educational affordances of AR by combining technology, pedagogy, and gaming perspectives. The review demonstrated that mobile AR can deliver context-sensitive content and enhance engagement, with notable success in public adoption through games such as Pokémon Go. However, the solution's limitation lies in its broad, survey-style focus; it identifies guidelines rather than providing an implementable framework. Our project extends this by delivering a concrete, event-driven AR cube system tied to cloud progression.

Amanatidis (2022) conducted a focused literature review of 78 studies on AR in education and AR serious games, exploring implementation, evaluation, and theoretical underpinnings [12]. The review emphasized cognitive benefits, increased motivation, and immersive engagement, while also highlighting critical barriers such as teacher readiness, infrastructure constraints, and curriculum alignment. It proposed evaluation of rubrics and frameworks to assess AR learning effectiveness across disciplines like science, mathematics, and language studies. While comprehensive in identifying challenges and future directions, this methodology is limited by its secondary analysis approach. Our system advances beyond review by implementing a functional AR cube prototype with Firebase integration, directly addressing deployment and user progression tracking.

Li et al. (2017) conducted a literature review of 26 AR learning games to analyze their effects on student learning outcomes, motivation, and social interaction [13]. The study classified learner groups, subjects, and environments, finding that AR games generally improved performance, engagement, and collaboration. Common design elements included quizzes, puzzles, goal setting, and 3D models. Evaluation methods relied on pre/post-tests, observations, and surveys, showing that AR games fostered enjoyment and teamwork. However, the review was limited to secondary synthesis and did not examine retention effects or implementation challenges in depth. Our project extends these findings by deploying a practical AR cube system with event-driven unlocking and cloud integration.

6. CONCLUSIONS

While the proposed AR cube system demonstrates promise in combining Vuforia-based tracking with Firebase authentication and cloud storage, several limitations remain. First, tracking reliability is highly dependent on environmental lighting and cube visibility; although accuracy was acceptable under normal conditions, performance declined in low light. This could be mitigated by integrating adaptive brightness detection or combining Vuforia with AR Foundation

for sensor fusion. Second, network latency poses challenges for seamless media playback. Current performance is strong under WiFi or LTE, but slower connections result in noticeable delays [14]. Implementing caching strategies, adaptive streaming, or lightweight media previews could enhance responsiveness. Third, data consistency in unlocking events remains a concern, as connectivity drops may cause sync errors. Employing Firestore transactional writes and redundancy mechanisms would reduce risk. Finally, user studies are limited in scale. Larger deployments in classrooms or museums would provide stronger validation, especially regarding long-term engagement and educational effectiveness.

This research shows that combining cube-based AR tracking with cloud-backed event unlocking can create immersive, engaging, and persistent learning experiences. By linking physical interactions to digital rewards, the system demonstrates a scalable framework for interactive education. Future improvements will strengthen its robustness, ensuring broad applicability in learning and training contexts.

REFERENCES

[1] Chang, George, Patricia Morreale, and Padmavathi Medicherla. "Applications of augmented reality systems in education." Society for Information Technology & Teacher Education International Conference. Association for the Advancement of Computing in Education (AACE), 2010.

[2] Moroney, Laurence. "Using authentication in firebase." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017. 25-50.

[3] Moroney, Laurence. "Cloud storage for firebase." The definitive guide to firebase: build android apps on google's mobile platform. Berkeley, CA: Apress, 2017. 73-92.

[4] Carmigniani, Julie, and Borko Furht. "Augmented reality: an overview." Handbook of augmented reality (2011): 3-46.

[5] Aughey, Robert J. "Applications of GPS technologies to field sports." International journal of sports physiology and performance 6.3 (2011): 295-310.

[6] Chaudhary, Meenu, et al. "Leveraging Unity 3D and Vuforia Engine for Augmented Reality Application Development." 2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS). IEEE, 2023.

[7] Sarkar, Swagato. "The unique identity (UID) project, biometrics and re-imagining governance in India." Oxford Development Studies 42.4 (2014): 516-533.

[8] Nadales, David Cantón. Build Your Own Metaverse with Unity: A practical guide to developing your own cross-platform Metaverse with Unity3D and Firebase. Packt Publishing Ltd, 2023.

[9] Jiménez Fernández-Palacios, Belen, et al. "ARC ube—The Augmented Reality Cube for Archaeology." Archaeometry 57 (2015): 250-262.

[10] Kesavan, Ram, et al. "Firestore: The nosql serverless database for the application developer." 2023 IEEE 39th International Conference on Data Engineering (ICDE). IEEE, 2023.

[11] Laine, Teemu H. "Mobile educational augmented reality games: A systematic literature review and two case studies." Computers 7.1 (2018): 19.

[12] Amanatidis, N. "Augmented reality in education and educational games-implementation and evaluation: a focused literature review." Computers and Children 1.1 (2022): 1-11.

[13] Li, Jingya, et al. "Augmented reality games for learning: A literature review." International Conference on Distributed, Ambient, and Pervasive Interactions. Cham: Springer International Publishing, 2017.

[14] Astély, David, et al. "LTE: the evolution of mobile broadband." IEEE Communications magazine 47.4 (2009): 44-51.

[15] Grahn, Ivar. "The vuforia sdk and unity3d game engine: Evaluating performance on android devices." (2017).