SAPIENZA
UNIVERSITÀ DI ROMA

ShARED: A Mobile Application for Designing, Sharing and Experiencing Marker-Based Augmented Reality Environments

Faculty of Information Engineering, Informatics, and Statistics
Master Degree in Engineering in Computer Science

Valerio Di Stefano
ID number 1898728

Supervisor
Roberto Beraldi

Academic Year 2023/2024

Thesis defended on 30 January 2025
in front of a Board of Examiners composed by:

Prof. Massimo Mecella (chairman)
Prof. Laura Astolfi
Prof. Roberto Beraldi
Prof. Silvia Bonomi
Prof. Fabrizio Silvestri
Prof. Luca Iocchi
Prof. Giuseppe Santucci

ShARED: A Mobile Application for Designing, Sharing and Experiencing Marker-Based Augmented Reality Environments
Experimental Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: distefano.1898728@studenti.uniroma1.it

Alla mia famiglia, ai miei amici, a me

Abstract

Introduction Augmented Reality (AR) has emerged as a transformative technology, bridging the gap between the digital and physical worlds by overlaying virtual content onto real environments, but the complexity of AR development and the reliance on specialized hardware or desktop software have limited its accessibility to non-technical users. This thesis introduces ShARED, a mobile application that democratizes AR creation by enabling users to design, share, and experience marker-based AR environments directly on their smartphones. The app aims to lower the entry barriers for AR adoption, providing an intuitive and user-friendly platform suitable for diverse applications and audiences.

Methods The development of ShARED involved leveraging Unity's AR Foundation framework and various marker-based AR techniques, employing computer vision algorithms for marker detection, pose estimation, and content tracking. A fully modular and mobile-centric design was implemented, integrating an in-app editor for creating and experiencing AR environments as well as features for sharing them with other users. The prototype was tested to evaluate its accuracy, speed, and performance optimizations, comparing different image tracking algorithms and optimization methods to assess their effectiveness in real-time AR applications.

Results The experimental evaluation of ShARED demonstrated its effectiveness in various use cases: the app's accuracy and speed were tested against different image tracking algorithms, with the AR Foundation library achieving the highest accuracy/speed trade-off and tracking performances. Ad-hoc solutions such as ORB detection and sensor-based tracking achieving better results in specialized detection and tracking tasks. Optimization methods, such as distributed detection and user-triggered marker detection, showed promising results in improving the app's performance and scalability. The results underscored ShARED's potential to redefine how AR environments are created and shared, expanding the reach of AR technology to a wider audience.

Discussion The results underscore ShARED's potential to redefine how AR environments are created and shared. By focusing on accessibility and customization, the app expands the reach of AR technology to a wider audience. Future work could explore the integration of markerless AR capabilities, generative AI tools for content creation, and adaptation to emerging AR and VR hardware, such as smart glasses. These advancements could further enhance the app's functionality and broaden its applicability across different domains. ShARED represents a significant contribution to the field of AR, laying the groundwork for innovations that connect digital content with real-world experiences in intuitive and impactful ways.

Contents

Chapter 1

Introduction

Augmented Reality (AR) has emerged in recent years as a transformative technology, blurring the lines between the digital and physical worlds. AR overlays digital content onto the real world, enhancing our perception and interaction with our surroundings, making it a versatile means to improve the productivity and efficiency of various industries, including healthcare, retail, gaming, education, marketing, and entertainment. Despite its potential benefits and various applications, AR development tools often require specialized skills and expensive hardware, limiting accessibility for non-experts and small businesses. This limitation has hindered the widespread adoption and exploration of AR technology in various fields where it could be beneficial, and the friction in the development process of AR experiences has been a significant barrier to entry for many potential users, most notably in the sectors where expertise in AR development is not a core competency, such as education, entertainment, or everyday consumer applications. Existing AR development platforms and tools often require users to have a deep understanding of programming languages, 3D modeling, and computer vision, which can be daunting for beginners and non-technical users. Even with the availability of more user-friendly AR development tools, such as no-code solutions or WYSIWYG editors, the creation of AR experiences still requires users to have access to a computer and be familiar with the concept of installing custom-built applications on their devices. This can be a significant barrier for users who are not tech-savvy or do not have access to a computer, limiting the potential audience for AR experiences. Moreover, the need for specialized hardware in some applications, such as AR glasses or XR headsets, further complicates the development process and limits the potential audience for AR experiences.

This thesis work addresses the challenge of democratizing AR technology by introducing ShARED, a mobile application that empowers users to design, share, and experience marker-based AR environments directly from their smartphones, without the need for specialized hardware, programming knowledge, or any other external device or tool, integrating everything into a single application.

ShARED distinguishes itself from traditional AR creation platforms by offering a purely mobile, code-free interface, eliminating the need for complex software or programming knowledge, while simultaneously providing a user-friendly approach to AR design, allowing individuals with varying technical backgrounds to easily create and customize AR environments and seamlessly sharing and experiencing them with other users.

The chapters of this thesis discuss and introduce the features and functionalities of the ShARED app (Chapter 2), followed by the technical aspects, design decisions and development challenges behind all of it's core components (Chapter 3), and ultimately the insights gained from testing and comparing the accuracy, efficiency and usability of multiple approaches and technologies implemented for the various application components (Chapter 4). The thesis concludes with a final discussion about possible use cases and future directions of the ShARED app (Chapter 5).

1.1 Introduction to Augmented Reality

Augmented reality (AR) is a technology that superimposes computer-generated images onto a user's view of the real world, creating a composite view that enhances the user's perception of reality. AR is distinct from virtual reality (VR)[6], which immerses users in completely virtual environments. Unlike VR, AR complements the real-world environment by overlaying digital objects onto it.

One of the key components of AR is the ability to track the user's position and orientation in real-time, allowing the digital content to be accurately aligned with the real world. This tracking can be achieved through various methods, including marker-based AR (which uses predefined images or markers in the real world to trigger and anchor digital content) and markerless AR (which relies on sensors and geographical position to determine the digital content to display in the

final AR environment).

AR technology can be implemented on a variety of devices, including smartphones, tablets, head-mounted displays (HMDs), smart glasses, and even projection systems. The choice of device depends on the specific application and the desired level of immersion.

AR has a wide range of existing or potential applications[2], including:

- Education and Training: AR can be used to create interactive learning experiences, simulations, and training programs in various fields, such as science, engineering, and healthcare.
- Commerce: AR is increasingly being used in retail to enhance product previews and provide virtual try-on experiences, or in marketing to create engaging and interactive campaigns.
- Industrial Manufacturing: AR can assist with tasks such as assembly, maintenance, and quality control by providing workers with real-time information and guidance.
- Healthcare: AR is being explored for applications in surgery, medical training, and patient rehabilitation.
- Entertainment: AR can be used to create immersive games, interactive art installations, and live events.

1.1.1 Extended Reality Terminology: AR, VR, MR and XR

Extended reality (XR) is an umbrella term encompassing various technologies that blend the real and virtual worlds[5]. Here's a breakdown of the key terms:

- Augmented reality (AR): AR overlays digital content onto the real world, enhancing the user's perception of reality. AR experiences are typically viewed through devices like smartphones, tablets, headsets or smart glasses. AR complements the real world, adding digital elements to it.
- Virtual Reality (VR): VR immerses users in fully simulated digital environments. VR experiences require the use of headsets that block out the real world, replacing it with a computer-generated environment. VR replaces the real world with a simulated one.

- Mixed reality (MR): MR combines elements of both AR and VR. MR experiences involve the interaction of real-world and digital objects, creating environments where physical and digital elements coexist and interact in real-time. MR allows users to interact with both the real and digital worlds simultaneously.
- Extended reality (XR): XR is the overarching term encompassing AR, VR, and MR. It refers to any technology that extends or alters our perception of reality, blurring the lines between the physical and digital worlds. XR ultimately combines or mirrors the physical world with a "digital twin world" to create immersive experiences.

1.2 Marker-based AR Experiences

Marker-based AR uses visual cues, or markers, to trigger and place augmented content in the real world. The markers can be simple patterns, like QR codes and ArUco markers, or unique "natural feature" images that the AR system can recognize. Marker-based AR relies on computer vision algorithms or machine learning and deep learning to detect and track markers in the user's real world environment.

Marker-based AR generally follows these steps:

- Marker Detection: The AR app uses the device's camera to scan the environment. When a marker is detected, it is isolated from the scene.
- Pose Estimation: The app calculates the position and orientation of the marker in the real world. It does so by analyzing the marker's perspective distortion in the camera image, using its known real-world size as a reference.
- Content Rendering: The app superimposes the augmented content onto the detected marker based on its calculated pose.

Marker-based AR has several advantages:

- Simplicity: It is relatively easy to implement, especially using AR SDKs.
- Accuracy: It provides accurate positioning and tracking of augmented content if the marker is completely or partially visible.

- Control: It allows developers to have more control over the placement and appearance of the augmented content, anchored to a specific marker.

Some examples of marker-based AR applications include:

- Interactive Educational Materials: Textbooks with markers that activate 3D models, animations, or videos.
- Product Packaging and Marketing: Product boxes with markers that show product demonstrations.
- Museum Exhibits: Markers near artifacts that show information about the exhibits.
- Gaming: Board games with markers that trigger animations or special effects.

Despite its advantages, marker-based AR also presents limitations:

- Marker Dependency: The AR experience is tied to the presence of markers, which can limit the user's freedom of movement.
- Marker Occlusion: If the marker is partially or fully occluded, the AR content may not render correctly.
- Marker Recognition: The AR system may not recognize markers in low-light conditions, at a distance, or at an angle.
- Marker Tracking: The AR system may struggle to track markers accurately if the camera moves quickly or the marker is far away.

Markerless AR vs. Marker-based AR

The counterpart of marker-based AR is markerless AR, which relies on sensors and computer vision algorithms to track the user's position and orientation without the need for markers. Markerless AR is more challenging to implement but offers greater freedom of movement and interaction. Markerless AR is often used in applications that require spatial understanding, such as indoor navigation, object recognition, and scene reconstruction[(8)].

Markerless AR systems use a combination of sensor data, such as camera images, GPS, and IMU readings, to estimate the user's position and orientation. They may also use machine learning algorithms, such as SLAM (Simultaneous Localization and Mapping), to create a 3D

map of the environment and track the user's movements. Markerless AR systems can be more computationally intensive than marker-based AR systems, as they need to process large amounts of sensor data and perform complex calculations in real-time.

Both marker-based and markerless AR have their strengths and weaknesses, and the choice between them depends on the specific application requirements. Marker-based AR is well-suited for applications that require precise tracking and alignment of digital content with physical objects, while markerless AR is better suited for applications that require spatial understanding and environmental interaction.

1.2.1 Binary vs. Natural Feature Markers

Marker-based AR experiences rely on distinct visual cues (markers) to trigger augmented content, which can be:

- Natural Feature Markers: These markers are images captured from the real world, like photographs, book covers, or artwork. They rely on intricate patterns and unique features within the image itself for detection and tracking. Feature extraction algorithms like SIFT, SURF, or ORB are commonly used to identify and match key points in natural feature markers.
- Binary Markers: These markers consist of simple, easily recognizable patterns, often black and white, like QR codes, AR tags or ArUco markers[9]. They are designed for robust detection and decoding, even under varying lighting and orientations. Binary markers simplify the detection process as their unique patterns are easily distinguishable.

While binary markers are efficient and reliable for AR applications, they lack personalization and flexibility (users are limited to using pre-defined markers), and make it mandatory for users to have access to real-world representations of such markers.
Natural feature markers, on the other hand, offer greater customization and creativity (users can choose any image as a marker), enabling users to create personalized AR experiences using their own photos or artwork, but can also be more challenging to detect and track due to the complexity and variability of the images (often requiring a preprocessing step to extract features that will be used for detection and tracking, and therefore demanding more computational resources).

Chapter 2

ShARED Introduction and Features

This chapter provides a detailed description of the ShARED mobile application from the user's point of view, focusing on its core functionalities, user interface, and user experience. In particular, the chapter outlines the three key features of the app: creation of AR environments, sharing of environments with other users, and experiencing AR environments created by oneself or others.

The chapter also highlights existing AR applications and tools that have a similar focus on creating and sharing AR content, emphasizing the unique aspects of ShARED that set it apart from other solutions.

For an in-depth explanation of the app's features implementation and design decisions, see Chapter 3.

2.1 App Introduction

The increasing availability and adoption of mobile devices equipped with cameras and sensors opened up a world of possibilities in the domain of augmented reality (AR), allowing users to superimpose computer-generated images on top of their view of the real world.

Recent advancements in XR technologies have made AR more accessible and user-friendly, allowing individuals to experience AR content without the need for specialized hardware or technical expertise. This led to the idea for an innovative mobile application that empowers users to design, share, and explore marker-based AR environments directly from their smartphones or tablets: ShARED, short for "Shared

Augmented Reality Environments Designer," is a mobile app that aims to democratize the creation and sharing of AR content, making it accessible to a wide variety of users.

The ShARED mobile application is a marker-based AR app, meaning it uses images, referred to as "markers," to trigger the display of augmented content when the device's camera detects them. The innovative feature of ShARED is its accessibility and user-friendliness, allowing users without prior experience in AR development to create and share their AR experiences directly from their mobile devices. This is achieved through an intuitive in-app editor that enables users to easily define marker images via their smartphone's camera, associate them with various types of augmented content, and customize the display of these elements within the AR environment.

The versatile nature of ShARED makes it suitable for a wide range of applications, including education, entertainment, and personalization of learning experiences. Educators can leverage the app to create interactive learning materials that engage students and enhance their understanding of complex concepts. Students can explore subjects in a more visual and interactive manner, gaining deeper insights and fostering curiosity. Individuals can personalize their learning journeys by creating AR experiences tailored to their specific needs and interests, transforming passive consumption of information into active engagement and exploration, like note taking through annotations shown directly on the page of a book, or creating interactive flashcards with 3D models that come to life when scanned with the app.

2.2 App Features Overview

The ShARED app revolves around three core functionalities:

- Designing and creating AR experiences: Users can create marker-based AR environments by defining marker images and associating them with various types of augmented content, such as text, images, videos, and 3D objects.
- Sharing AR experiences with other users: Users can publish their AR creations online, making them accessible to a broader audience. Shared experiences can be public or private, allowing users to control the visibility and accessibility of their content.

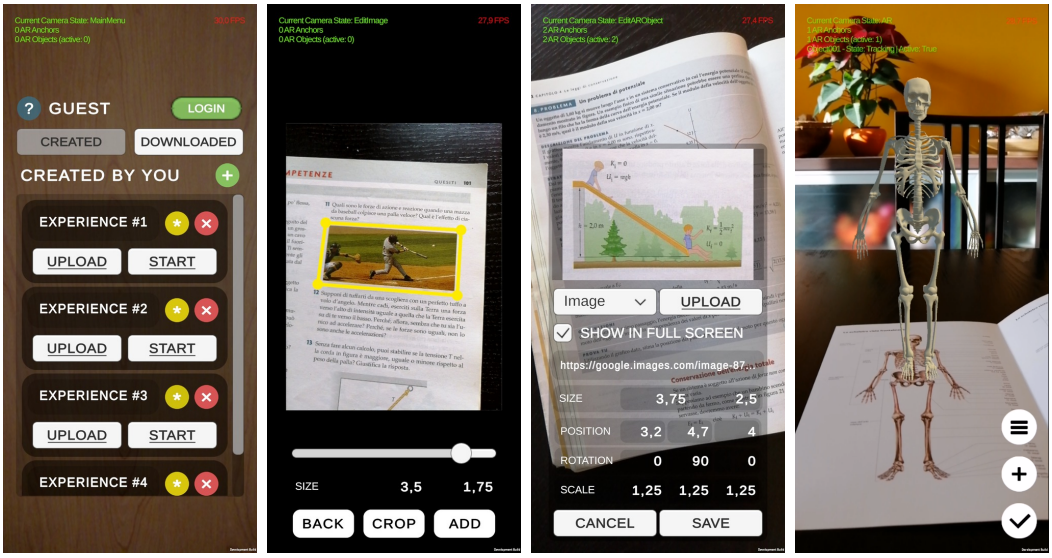


Figure 2.1 Screenshots of the ShARED app prototype.

- Experiencing AR environments created by oneself or other users in a local or shared environment: Users can explore and interact with AR environments created by themselves or others, leveraging the app's marker recognition and tracking capabilities to display augmented content in real-time, as well as the app's live-sharing features to engage in collaborative learning experiences or shared entertainment activities.

The remainder of this section provides an in-depth look at each of these three main functionalities, highlighting the key features and user interactions involved in the process.

2.3 Creation of AR Environments

ShARED empowers users to design their own marker-based augmented reality (AR) experiences directly from their smartphones, without requiring external software or technical expertise. The creation process is accessible through an intuitive in-app editor, allowing users to unleash their creativity and personalize their AR environments in just a few steps. This section delves into the features and steps involved in the AR experience creation phase from a user's perspective.

2.3.1 Initiating the Creation Process

To begin designing an AR experience, users navigate to the main editor menu of the ShARED app, where they can create a new, blank AR environment. Each environment acts as a container for a collection of markers and their associated augmented content. Once an environment is initialized, users can open it in "Edit Mode," a workspace tailored for adding and customizing the elements of the AR experience.

2.3.2 Adding Image Markers

Markers are central to ShARED's AR functionality. These are user-defined planar images that serve as anchors for augmented content. Users can seamlessly add new markers by taking a photo of a desired object or surface directly within the app. This photo becomes the basis for the marker. To ensure accurate recognition and functionality, users can refine the captured image by adjusting its perspective and cropping it.

The app provides an interactive tool for cropping, allowing users to define a polygonal cropping area around the marker image. Each corner of this polygon is movable, giving users fine control to ensure the cropped image aligns perfectly with the planar object. This step is crucial for creating a clear and distinct marker that the app can reliably recognize. Users can preview the adjusted image to confirm its alignment and re-crop it as needed to achieve the desired result.

After finalizing the marker image, users can specify its approximate real-world dimensions. This information enhances the app's ability to detect the marker and appropriately scale its associated augmented content.

2.3.3 Attaching Augmented Content

Once a marker is defined, users can enter a dedicated editing interface to attach augmented content. ShARED supports four types of augmented content:

- Text: Users can type and format textual content directly within the app. This feature is particularly useful for annotations, notes, or educational information.

- Images and Videos: Users can upload static images or videos from their device, which are displayed as floating 2D elements anchored above the marker in the AR environment.
- 3D Objects: Any 3D model, including animated ones, can be added. These models are rendered in three-dimensional space, creating an immersive experience.

The app's content editor offers a range of customization options to ensure the augmented content integrates seamlessly with the marker and the real-world environment. Users can:

- Adjust the position of the content relative to the marker's center using X, Y, and Z coordinates.
- Modify the rotation by specifying angles for each axis (pitch, yaw, and roll).
- Scale the content uniformly or non-uniformly by adjusting the size along each axis (X, Y, Z).

2.3.4 Full-Screen Mode

In addition to embedding content spatially within the AR environment, users can opt for "Full-Screen Mode." This mode displays the augmented content (e.g., text, image, video, or 3D model) on the user's smartphone screen instead of overlaying it in the 3D space. Full-Screen mode is activated when the corresponding marker is detected. This feature is especially advantageous for content that benefits from detailed viewing without spatial constraints, such as instructional videos or intricate images. Here, the marker acts as a trigger, guiding users to access the content conveniently.

2.3.5 Finalizing the Experience

Users can continue adding markers and their corresponding augmented content to their environment until the design is complete. Each marker's content and parameters are saved within the environment, enabling users to revisit and edit them later. Once satisfied with their creation, users save the environment as a finalized package, ready to be shared or experienced.

2.3.6 Managing Multiple Environments

ShARED allows users to create and manage multiple independent AR environments, each containing its unique collection of markers and augmented content. This flexibility supports diverse applications, from educational tools and personal projects to collaborative and professional use cases. Users can switch between environments, edit existing ones, or start fresh projects as desired.

2.4 Sharing of AR Environments

Once an experience is ready, meaning the user has created the experience itself, defining a collection of anchor images with their related augmented content to be displayed inside the app, the user will be able to make the experience available for other users of the app. This process ensures that creations can be widely accessed, shared, and experienced by a diverse audience.

2.4.1 Authentication Process for Sharing

To share an experience, the user must first authenticate within the app. Authentication is straightforward and designed for convenience:

- The user provides an email address.
- A "One Time Password" (OTP) is emailed to the user.
- The user enters the OTP in the app to complete the login process.

Future versions of the app may include alternative authentication methods, such as traditional username and password or OAuth logins, providing more flexibility to users. Once authenticated, users can customize their public profile, deciding how other users perceive them. Users who value anonymity can opt to hide their identity while still sharing their experiences.

2.4.2 Uploading the AR Experience

After authentication, users can proceed to upload their AR experiences. The process involves the following steps:

- Define a name for the AR experience to make it easily identifiable.

- Optionally, provide a detailed description of the experience, highlighting its features or intended use.
- Click the "Upload" button to publish the experience online.

Once the upload is initiated, the app prompts the user to choose the sharing preferences for the experience. This step ensures that users have full control over how their creations are distributed.

2.4.3 Defining Sharing Preferences

Users can select from two distinct sharing modes:

- Public Sharing: In this mode, the experience becomes searchable by name and/or the creator's profile. Other users can view the creator's collection of shared experiences and select experiences they find interesting.
- Private Sharing: This mode restricts access to the experience, making it visible only to users who possess a unique code. The unique code is generated automatically during the upload process and ensures that the experience is not searchable or visible in the creator's profile.

These sharing options cater to both users who wish to broadcast their creations widely and those who prefer or need to share their content selectively with a specific audience (e.g. professors sharing an AR experience with their classroom to augment the content of a textbook).

2.4.4 Accessing Shared Experiences

Beyond sharing, the app facilitates seamless access to experiences created by others. Users can discover new environments in two main ways:

- Search for public experiences or creators using names or usernames. This feature allows users to browse a wide array of publicly available AR environments.
- Input a unique code to access private experiences shared with them. This method ensures that access to restricted content remains controlled and secure.

Once an experience is selected, users can download it locally to their smartphones. The download includes all associated augmented content, ensuring a complete and immersive AR experience.

2.4.5 Ensuring Accessibility and Usability

The app's sharing functionality is designed to maximize accessibility. Downloaded experiences are stored on the user's device, enabling offline access and reducing dependency on network availability during use. This feature is particularly beneficial for educational or professional settings where reliable internet connectivity may not always be available.

2.5 Experiencing AR Environments

Once an AR environment is either created by a user or downloaded from the shared experiences available online, it can be explored and interacted with using the ShARED app. This phase offers users an immersive and engaging way to visualize augmented content integrated with their real-world surroundings.

2.5.1 Discovering AR Content

The first step in experiencing an AR environment involves utilizing the smartphone's camera to scan for markers within the real-world environment. These markers, when detected by the app, act as anchors in the 3D space, triggering their associated augmented content.

The process is designed to be seamless and intuitive:

- Users launch the AR experience mode from the app's main interface.
- The camera scans the surroundings, searching for predefined markers embedded in the environment.
- Once a marker is detected, the app overlays the corresponding augmented content either spatially in the real-world view or as full-screen content.

2.5.2 Interacting with Augmented Content

The app supports a wide range of interactions with the augmented content. Users can manipulate 3D objects, media files, and other elements to suit their preferences. Interaction options include:

- Moving Content: Users can reposition augmented objects within the 3D space by dragging them across the screen.
- Scaling Content: Pinch-to-zoom gestures allow users to adjust the size of augmented objects, making them larger or smaller as needed.
- Rotating Content: Augmented objects can be rotated around their axes using intuitive touch gestures, ensuring precise alignment and orientation.

For certain types of content, such as videos or animated 3D models, users gain additional controls:

- Playback Controls: Users can start, pause, resume, or loop videos and animations directly from the app interface.
- Timeline Adjustments: A slider enables users to control the playback timeline, allowing them to focus on specific segments of animations or videos.

2.5.3 Full-Screen Viewing Mode

??
In addition to spatial overlays, the app offers a full-screen mode for experiencing augmented content. When a marker is detected, users can opt to display the associated augmented content on their smartphone screens without embedding it into the real-world view. This mode is particularly beneficial for:

- Viewing detailed images or videos that require focused attention.
- Reading textual information in a more accessible format.
- Exploring complex 3D models in isolation from their physical surroundings and without worrying about controlling their smartphone's camera.

2.5.4 Collaborative and Live-Sharing Sessions

One of the main features of the ShARED app is its support for live-shared, multi-user sessions. This capability allows multiple users to experience the same AR environment simultaneously within a shared physical space. The functionality works as follows:

- Hosting a Session: A user assumes the role of a host, controlling the augmented content displayed in the shared space.
- Connecting Participants: Other users join the session through a local network or Bluetooth connection, synchronizing their devices with the host.
- Real-Time Updates: Changes made by the host to the augmented content, such as repositioning or playback adjustments, are reflected on all connected devices in real-time.

This feature is particularly useful in educational settings, where a teacher can guide students through interactive 3D models or animations, ensuring everyone observes the same content simultaneously. Another application is collaborative work, where team members can review and discuss 3D models or visualizations together, enhancing communication and understanding.

2.5.5 Personalized and Offline Access

Users retain control over their experiences, with the ability to personalize content interactions and access environments offline. Downloaded AR environments are stored locally on the user's device, enabling:

- Continuous access to experiences without requiring internet connectivity.
- Greater flexibility for use in remote or bandwidth-limited locations.
- Long-term storage of favorite environments for repeated exploration.

2.6 Existing Tools and Platforms for AR Experience Creation

Augmented Reality (AR) tools and platforms have evolved significantly, offering various methods to create marker-based or markerless AR experiences. These tools can be broadly categorized into six distinct types, each catering to different levels of expertise, goals, and functionalities. This section provides an overview of these categories followed by an in-depth analysis of their features and limitations.

2.6.1 Overview of AR Creation Methods

AR tools can be classified as follows:

- Coding from Scratch: Developing AR applications entirely through custom code, offering complete control and flexibility.
- General Graphic Engines or AR SDKs: Leveraging platforms like Unity, Unreal Engine, Godot Engine or SDKs for AR development like ARKit and ARCore to build AR experiences.
- Dedicated AR Development Tools: Using tools such as Vuforia, Wikitude, and EasyAR, designed specifically for AR application development.
- WYSIWYG AR Editors: Platforms like ZapWorks, Blippbuilder, HP Reveal and Adobe Aero that provide visual, drag-and-drop interfaces for AR creation.
- AR Content Creation Platforms: Utilizing platforms such as Metaverse, 8th Wall, and Torch AR for simplified AR content creation and publishing.
- Direct AR Apps on End-User Devices: Apps that allow users to create and experience AR content directly on smartphones or tablets.

2.6.2 Coding from Scratch

Creating AR applications from scratch involves writing code to implement AR functionalities without relying on predefined libraries or frameworks. This method offers maximum customization and flexibility, allowing developers to:

- Implement advanced features tailored to specific use cases.
- Optimize performance for specific hardware or software constraints.
- Integrate AR functionalities with other custom applications seamlessly.

However, this approach requires substantial expertise in computer vision, 3D graphics, and hardware programming. Additionally, development timelines are often prolonged, making this method less accessible for non-technical users or rapid prototyping.

2.6.3 General Graphic Engines or AR SDKs

Game engines like Unity, Unreal Engine and Godot, or AR SDKs like ARKit and ARCore, provide comprehensive tools and software development kits (SDKs) for AR application development. These solutions offer:

- Prebuilt libraries for marker detection, image tracking, and pose estimation.
- Integration with 3D modeling and animation tools.
- Cross-platform compatibility, allowing developers to deploy applications on multiple devices.

While these tools simplify AR development compared to coding from scratch, they still require proficiency in programming and familiarity with the chosen engine or SDK. The need to export and deploy applications adds another layer of complexity, particularly for mobile-centric use cases.

2.6.4 Dedicated AR Development Tools

Dedicated tools like Vuforia, Wikitude, and EasyAR are designed explicitly for AR development, offering:

- Advanced marker-based and markerless tracking capabilities.
- Support for integrating multimedia content such as videos, 3D models, and animations.
- Cloud-based image recognition and data storage options.

These platforms reduce the technical barriers to entry by providing specialized features, but they often come with licensing fees and limited customization options. Moreover, the reliance on external tools may restrict flexibility for developers seeking more control over their applications, while their "multi-stop" design process (first develop the AR application on a computer, then build it, export it and deploy it on a mobile devices) may still pose as an entry barrier for most users.

2.6.5 WYSIWYG AR Editors

What-You-See-Is-What-You-Get (WYSIWYG) editors such as ZapWorks, Blippbuilder, HP Reveal or Adobe Aero enable users to create AR experiences through visual interfaces without coding.

Key benefits include:

- Drag-and-drop interfaces for intuitive design.
- Prebuilt templates for common AR scenarios.
- Rapid prototyping and deployment of AR content.

However, these platforms typically offer limited customization and are constrained by the predefined templates and asset libraries. Advanced functionalities and integration with external systems may also be unavailable. Moreover, they still present a "multi-stop" design process, whose limitations were introduced above.

2.6.6 AR Content Creation Platforms

Platforms like Metaverse, 8th Wall, and Torch AR provide streamlined solutions for creating and sharing AR content.

Their features include:

- Web-based interfaces for easy access and collaboration.
- Tools for publishing AR experiences directly to the web or mobile apps.
- Support for both marker-based and markerless AR.

These platforms prioritize simplicity and accessibility but often limit users to predefined workflows and asset libraries. Additionally, the reliance on web technologies may result in performance trade-offs compared to native applications.

2.6.7 Direct AR Apps on End-User Devices

AR apps designed for end-user devices allow content creation and interaction directly on smartphones, tablets, or headsets.

These apps typically:

- Eliminate the need for external hardware or software tools.
- Offer user-friendly interfaces tailored for non-technical users.
- Enable real-time testing and iteration within the same environment.

While these apps enhance accessibility and convenience, introducing a "single-stop" design process (development and deployment is done in-app), they often rely on prebuilt asset libraries and limited editing functionalities. Additionally, most existing solutions focus on markerless AR, restricting their applicability for marker-based use cases.

2.6.8 Comparative Analysis for Existing Tools and Platforms

Each category of AR tools and platforms offers unique strengths and limitations. While coding from scratch and using general engines provide unmatched flexibility, they demand significant expertise and resources. Dedicated tools and WYSIWYG editors strike a balance between functionality and ease of use but may sacrifice customization. Content creation platforms and direct AR apps prioritize accessibility and rapid prototyping but often lack the depth required for complex applications. Furthermore, all of the existing fully-mobile AR editors focus on markerless AR (thus don't provide any marker definition feature) and lack customization option for the AR content, being limited to predefined templates and asset libraries.

The ShARED app represents a novel approach to AR content creation, sharing, and experiencing, offering a comprehensive and user-friendly platform for designing, sharing, and exploring marker-based AR environments directly from mobile devices with a focus on fully customizable AR environment content.

2.7 ShARED in the Context of Existing Tools and Platforms

The majority of the AR tools and platforms discussed in the previous sections are "multi-stop" desktop applications. These tools require users to have a strong familiarity with desktop computing operating systems, the specific paradigms of the chosen tool, and the processes involved in exporting and deploying applications to mobile or dedicated XR devices. This reliance on desktop environments introduces several barriers to accessibility, particularly for users without technical expertise or access to powerful computing hardware.

The content of this section highlights the main challenges faced by desktop-centric AR tools and platforms, as well as the limitations of existing fully-mobile AR editors and the innovative aspects of ShARED that set it apart from the previously discussed existing solutions.

2.7.1 Desktop-Centric Challenges

Most AR tools, apart from the last category discussed (direct AR apps on end-user devices), share common characteristics that limit their accessibility and ease of use:

- Complex Interfaces: These tools often have steep learning curves, requiring users to invest significant time in understanding their functionalities and workflows.
- Dependency on Desktop Operating Systems: The reliance on desktop platforms necessitates familiarity with specific operating systems and their associated file management and application workflows.
- Exporting and Deployment: Users must navigate the technical complexities of exporting projects and deploying them to mobile devices or XR hardware, which can be a daunting task for non-technical users.

These factors make traditional AR creation tools less accessible to casual users or those seeking quick, intuitive ways to develop and share AR experiences.

2.7.2 Limitations of Pre-Made Asset Libraries and Markerless AR

Another significant limitation of most existing tools, including desktop-based and mobile platforms, lies in their reliance on pre-made asset libraries and their focus on markerless AR. These constraints hinder the creative and functional possibilities for users:

- Pre-Made Asset Libraries: Many platforms restrict users to selecting virtual content from in-app libraries of pre-designed assets. While this approach simplifies the creation process, it limits personalization and creativity, as users cannot easily incorporate their own unique content.

- Markerless AR: Existing tools predominantly support markerless AR, where virtual content is anchored to general locations in the real world, such as specific points on a floor or wall. This method lacks the precision and flexibility of marker-based AR, which allows content to be tied to specific objects or images, enabling more contextual and interactive experiences.

2.7.3 ShARED: A Fully-Mobile and Fully Customizable Marker-Based Solution

ShARED distinguishes itself from other existing solutions by addressing these limitations directly. It offers a fully-mobile platform that combines the familiarity and convenience of smartphones and tablets with advanced marker-based AR capabilities.

Key features of ShARED that set it apart from existing tools and platforms include:

- Mobile-Centric Design: Unlike desktop-dependent tools, ShARED operates entirely on mobile devices, eliminating the need for exporting and deployment workflows. This approach leverages users' familiarity with mobile interfaces, making the platform more intuitive and accessible.
- Marker-Based AR: ShARED enables users to create and experience AR environments using user-defined markers, providing full control over how and when augmented content is displayed. By allowing users to tie augmented content to specific objects or images, ShARED enhances the potential for personalization and interactivity.
- Customizable Content: The platform allows users to upload their own virtual assets, including text, images, videos, and 3D models, rather than relying on pre-made libraries. This feature fosters creativity and ensures that users can fully tailor their AR experiences to their specific needs and preferences.

By overcoming the limitations of traditional AR tools, ShARED provides a novel approach for AR environments creation, sharing, and experiencing. Its fully-mobile approach democratizes AR creation, enabling users of all technical skill levels to participate. Furthermore, its focus on marker-based AR and fully customizable content empowers

users to design highly personalized and contextually rich environments. These unique features position ShARED as a versatile and user-friendly platform for exploring the creative potential of augmented reality across diverse domains and applications.

Chapter 3

System Design and Implementation

This chapter presents a comprehensive overview of the development process for the ShARED application. The focus is on the methodologies and technologies employed to design and implement the app's prototype, which incorporates all the described core features, including image recognition, detection and tracking, other than cloud-based AR experiences sharing and multi-user live sharing functionalities.

The final section of this chapter provides an in-depth analysis of the image recognition, pose detection and tracking algorithms and techniques implemented in the ShARED app.

The next chapter will delve into the testing and evaluation of the app's core functionalities, along with the comparison of different approaches and optimizations for image recognition, pose detection, and tracking.

3.1 Overview of the Development Process

The work carried out for this thesis project aimed not only at presenting and discussing a fully-mobile marker-based AR application for designing, sharing, and experiencing AR environments, but also at implementing an actual working prototype of the app. The final minimum viable product (MVP) of the ShARED app was developed for Android smartphones and tablets, implementing all the core features described in the previous chapter, including designing and sharing AR environments, as well as live multi-user sharing of AR experiences.

This first section provides an overview of the entire development pro-

cess without going into technical details. The subsequent sections will delve into the technical aspects of the app's development, including the implementation of core features, backend server integration, technology selection, and performance optimization, for each of the core components of the ShARED app.

The development process involved several stages, each addressing specific technical and functional aspects of the ShARED app. The primary components of this process included:

- Prototype Design: Developing a clear blueprint for the app's architecture and functionalities, ensuring a structured approach to implementation.
- Technology Selection: Leveraging existing tools and libraries, including Unity, AR Foundation, and OpenCV, to streamline the development process while maintaining flexibility and scalability.
- Core Feature Implementation: Building and testing the app's main functionalities to achieve a reliable and engaging user experience.
- Backend Development: Establishing a dedicated server for the remote storage of AR environments and user data, alongside implementing network protocols for live sharing.
- Implementation, testing, and comparison of different approaches for each of the core features and components of the app.
- Performance Optimization: Conducting iterative testing and refinement to ensure efficient operation on mobile devices, particularly under resource constraints.

3.1.1 Prototype Development

The prototype was developed using the Unity engine, a versatile platform for creating interactive 3D and AR applications. Unity's integration with the AR Foundation library facilitated the implementation of advanced AR features. The AR Foundation library acted as a bridge to underlying AR technologies such as ARCore for Android devices, providing a unified interface for developers.

Unity Engine

Unity is a widely used cross-platform game engine that supports real-time 2D and 3D application development. Its core features include:

- A robust editor for designing, testing, and iterating applications efficiently.
- Extensive support for programming in C#, a language known for its performance and ease of integration in the Unity environment.
- A large ecosystem of plugins, assets, and community-driven resources to accelerate development.

AR Foundation Library

AR Foundation is a high-level framework that simplifies the integration of AR functionalities. It abstracts platform-specific details, providing a unified API for features such as:

- Image tracking to detect and track user-defined markers.
- Pose estimation to calculate the spatial position and orientation of tracked objects.
- Environment understanding to enable context-aware AR experiences.

This abstraction allowed the ShARED prototype to maintain cross-platform compatibility, supporting both Android and iOS devices with minimal code changes.

3.1.2 Core Functionalities

The app's core functionalities were implemented using a combination of Unity's tools and external libraries, with each feature undergoing extensive testing and refinement.

Image Recognition, Pose Detection, and Tracking

The app's image recognition and pose detection capabilities relied heavily on OpenCV, a powerful computer vision library. Various algorithms were tested and compared to identify the optimal solution for marker recognition and pose estimation. These included:

- ORB[13] (Oriented FAST and Rotated BRIEF): A fast and efficient algorithm suitable for mobile applications.
- SIFT[12] (Scale-Invariant Feature Transform): Known for its robustness in identifying features under varying conditions.
- SURF[14] (Sped-Up Robust Features): Optimized for speed while maintaining accuracy.
- AKAZE[10][11] (Accelerated-KAZE): A balance between speed and accuracy, particularly for mobile devices.

For tracking, custom solutions were explored that combined IMU sensor data, image processing, and sensor fusion techniques. Kalman filters were employed to improve the accuracy and stability of tracking by predicting marker positions and smoothing detected movements.

Server Integration

A dedicated simple backend server was developed to support the app's sharing and multi-user features using Python and Flask. The server provided:

- Secure storage of user-generated AR environments and associated data.
- Authentication via email-based one-time passwords, ensuring secure access.
- Request handling for AR environment storage and retrieval as well as user information updates.

Multiplayer and Multi-User Live Sharing

To enable multi-user live sharing of AR environments, Unity's multiplayer solution, Netcode for GameObjects, was integrated into the app. This framework facilitated real-time synchronization and interaction between multiple users within the same AR experience. Key features of the multiplayer implementation included:

- Real-time updates to shared AR environments, ensuring all users view consistent augmented content.
- Support for a host-client model, where one user acts as the host managing the session, while others join as clients.

- Efficient data synchronization to minimize latency and ensure smooth interactions, even on mobile devices with limited computational power.

3.1.3 Programming and Libraries

The development process for the ShARED app required the use of several programming languages and external libraries to implement, test, and refine its functionalities. These tools provided the necessary support for developing core features, backend systems, and testing environments. Below is an overview of the programming languages and libraries used:

- C#: The primary programming language used for implementing the app's functionalities within Unity. Its robust features and seamless integration with Unity's APIs made it ideal for handling the app's AR features and user interactions.
- AR Foundation and ARCore: These libraries were pivotal in enabling marker-based AR functionalities. AR Foundation served as a high-level abstraction layer, simplifying access to ARCore's robust image recognition, tracking, and environment understanding capabilities.
- OpenCV and its associated Unity library (OpenCVForUnity): OpenCV was used extensively for advanced image processing, including feature extraction and pose detection. The OpenCV for Unity wrapper facilitated the seamless integration of OpenCV's functionalities into Unity, allowing the app to leverage its powerful algorithms directly within the Unity environment using the C# language.
- Netcode for GameObjects: Unity's multiplayer networking solution, which enabled the implementation of the app's multi-user live sharing functionality. This library was essential for achieving real-time synchronization and ensuring consistent AR experiences across multiple devices.
- HTML and JavaScript: These languages were utilized for creating web-based custom made interface tools designed to assist in testing various methods and implementations of the app's core functionalities (for example the image labeling tool used for creating ground-truth data for testing image recognition, pose detection and tracking algorithms, discussed in Chapter 4).

- Python: Python was employed for processing testing data and analyzing performance metrics during the development phase. Its extensive libraries for data processing and visualization proved invaluable for evaluating and refining the app's functionalities.
- Flask: A lightweight web framework for Python, used to develop the backend server that supported the app's sharing and multi-user features. Flask provided a simple yet powerful platform for handling user requests, managing data storage, and ensuring secure communication between the app and the server.
- MatPlotLib: A Python library for creating visualizations and plots, utilized to create the graphs and charts that helped analyze the app's performance metrics and user interactions (most of which can be found in the next chapter of this thesis). MatPlotLib's flexibility and ease of use made it an essential tool for interpreting and presenting the data collected during testing and optimization.

3.2 Core Functionalities and Components

The development of the ShARED application followed a modular design approach to ensure flexibility, adaptability, and efficiency in testing various methodologies.

Each core functionality of the app was implemented as an independent component, functioning as a "black box" with defined inputs and outputs, ensuring that changes or upgrades to one module do not disrupt the overall functionality of the system. This modular architecture allowed seamless integration and replacement of individual components, enabling iterative testing and optimization while maintaining a fully operational prototype. The modular approach also supports scalability, as new features or improved methodologies can be integrated with minimal impact on the existing architecture.

The remaining of this section will first provide an overview of the core functionalities and components of the ShARED application, and then delve into more in-depth technical and implementation details of each module, including the implementation, testing, and optimization processes.

3.2.1 Overview of Core Functionalities

The core functionalities of the ShARED application can be grouped into distinct modular components, each designed to handle specific aspects of the AR environment creation, sharing, and experiencing process. These components are:

- Camera Module
  - Description: Captures the real-world scene and provides image data for further processing.
  - Input: Mobile camera sensor measurements.
  - Output: Captured camera image.
- Inertial Measurement Unit (IMU)
  - Description: Provides the pose and motion of the mobile device at the current time instant.
  - Input: Mobile position and motion sensor measurements.
  - Output: Device orientation, acceleration, and rotation vectors.
- Image Recognition Module
  - Description: Identifies the presence of markers in the environment by analyzing feature matches.
  - Input: Camera image and anchors' feature descriptors.
  - Output: Detected anchors list and matched feature keypoints.
- Pose Estimation Module
  - Description: Calculates the spatial position and orientation of detected markers in the current video frame.
  - Input: Matched feature keypoints, intrinsic matrix (camera parameters), and marker real-world sizes.
  - Output: Pose of each marker relative to the camera.
- Tracking Module
  - Description: Tracks the position and orientation of anchors over time with respect to the device frame.
  - Input: Camera image, initial anchor pose, and IMU data.
  - Output: Anchor pose and tracking status (tracking or lost).
- AR Objects Manager

  - Description: Manages the position, rotation, and scale of AR objects based on their associated anchor's pose.
  - Input: Associated anchor pose, AR object metadata, and default transform (position, rotation, scale).
  - Output: World space transform of each AR object.

- User Interaction Module
  - Description: Handles user interactions, such as translation, rotation, scaling, and playback controls for AR objects.
  - Input: User touch position and gestures, along with the current AR object state.
  - Output: Updated AR object transform or state.

- Serialization and Deserialization Modules
  - Description: Converts AR experience data into a byte stream for storage and sharing, and vice versa.
  - Input (Serialization): AR environment/experience data.
  - Output (Serialization): Bytes representation of the AR environment/experience.
  - Input (Deserialization): Bytes representation of an AR environment/experience.
  - Output (Deserialization): AR environment/experience data.

- Local Storage Module
  - Description: Stores user-created and downloaded AR experiences locally on the device.
  - Input/Output: Serialized experience data.

- Remote Storage Module
  - Description: Provides cloud-based storage for user data and shared AR experiences.
  - Input/Output: Serialized experience data.

- Multi-User Communication Module
  - Description: Synchronizes displayed AR objects across multiple devices in real-time.
  - Input: AR object transform updates and state updates for each connected device.

  - Output: Final transform and state of AR objects for each device.

- Server Module
  - Description: Handles requests for AR environment storage, retrieval, and synchronization among users.
  - Input: User requests and remote storage data.
  - Output: Server responses, including serialized AR experience data.

This modular architecture ensures that each component operates independently, with clear input and output definitions. Such a design facilitates the replacement, testing, and optimization of individual modules without affecting the overall functionality of the application. The modular approach also aligns with the goal of creating a flexible, scalable, and testable platform for marker-based AR experiences.

Figure 3.1 provides a high-level overview of the ShARED application's core modules and their interactions.

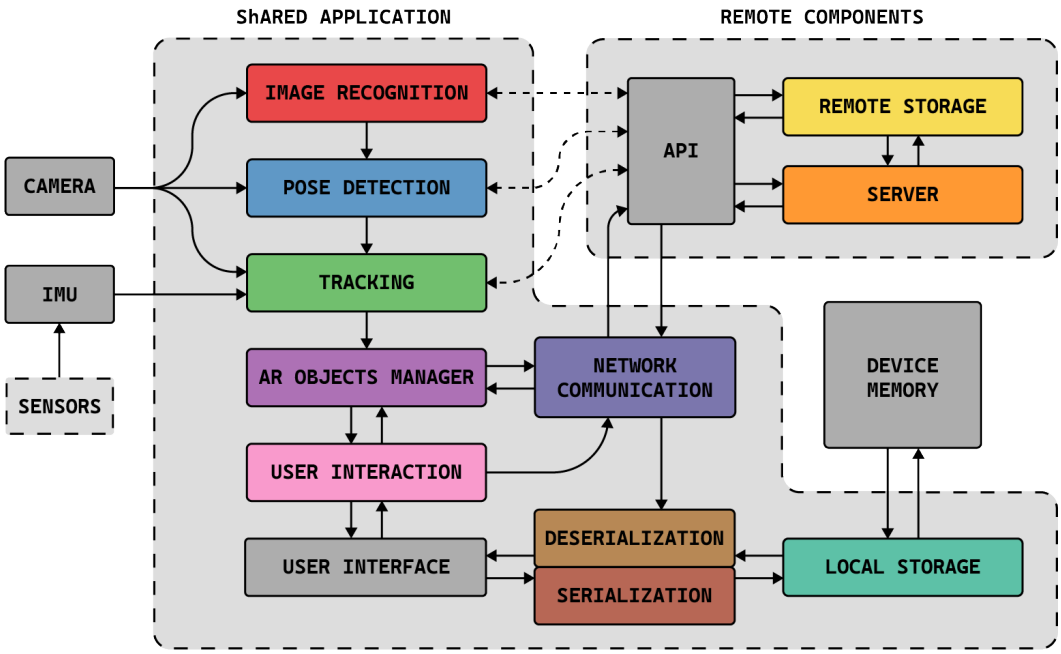


Figure 3.1 Overview of ShARED's core modules.

3.2.2 Camera and IMU Modules

The Camera and IMU modules interact directly with the Android architecture's exposed APIs to retrieve data critical for the application's functionalities. These components provide the foundational inputs for modules such as Image Recognition, Pose Estimation, and Tracking, enabling accurate and responsive augmented reality experiences.

Camera Module

The Camera Module captures real-time images of the surrounding environment using the mobile device's camera sensors. These images serve as the primary visual data for detecting markers and generating augmented content. The Camera Module ensures a steady stream of high-quality image frames, which are forwarded to the Image Recognition Module to identify user-defined markers and calculate their keypoints. These processed outputs are then utilized by the Pose Estimation Module to determine the spatial position and orientation of the markers relative to the camera.

IMU Module

The IMU Module leverages the device's motion and orientation sensors to deliver measurements such as acceleration, rotation, and orientation vectors. This information plays a crucial role in refining pose estimation and improving the reliability of the Tracking Module. By providing real-time data about the device's movement and orientation, the IMU Module helps stabilize marker tracking even under dynamic conditions, such as when the device is in motion.

3.2.3 Image Recognition and Pose Detection

The Image Recognition and Pose Detection modules are at the base of ShARED's marker-based AR functionalities. These modules work in tandem to identify user-defined markers in the real world and estimate their precise spatial position and orientation, enabling the accurate overlay of digital content onto the physical environment.

Image Recognition

The Image Recognition module is responsible for identifying the presence of pre-defined markers within the camera feed. This module

leverages computer vision techniques to analyze the incoming image stream and determine if any of the user-defined markers are visible. Once a marker is identified, its location within the image is determined, serving as an initial input for the Pose Estimation module.

Pose Detection

The Pose Detection module takes the identified marker from the Image Recognition module and calculates its precise 3D pose relative to the camera. This pose consists of both the marker's position (its location in 3D space) and its orientation (the direction it's facing).

Accurate pose estimation is crucial for ensuring that augmented objects are correctly aligned with the marker. This involves understanding the marker's position in relation to the camera and determining its rotation along three axes (pitch, yaw, and roll). The output of this module is a 6DOF (six degrees of freedom) pose, providing a complete representation of the marker's position and orientation within the 3D environment.

Implemented Image Recognition and Pose Detection Algorithms and Techniques

The ShARED application's modular approach allows for the testing and comparison of various algorithms for both image recognition and pose detection. This approach was crucial in evaluating the performance and suitability of different algorithms and methods for both marker image recognition and pose estimation.

The following provides an overview of the algorithms and techniques implemented and tested within the ShARED app, which will be discussed in more detail in the dedicated sections of this chapter, while an analysis of the experiments and tests for their performance and efficiency will be provided in the next chapter.

The prototype implementation of ShARED includes the testing and comparison of the following algorithms for image recognition and pose detection:

- AR Foundation/ARCore Default Solution: The application leverages the default image recognition and pose detection capabilities

provided by AR Foundation, which in turn utilizes platform-specific AR libraries like ARCore on Android devices. The specific algorithms employed by AR Foundation/ARCore are proprietary and not publicly disclosed, potentially involving a combination of traditional computer vision techniques and machine learning approaches. While generally efficient, these algorithms may not always provide the highest level of accuracy or robustness under challenging lighting conditions or when using natural feature markers.

- ORB[13] (Oriented FAST and Rotated BRIEF): ORB is a fast and efficient feature detection and description algorithm that is well-suited for mobile applications. It combines the FAST keypoint detector with a modified version of the BRIEF descriptor, achieving a good balance between speed and accuracy. While ORB can provide reasonably accurate pose estimation, it might be less robust than SIFT or SURF when dealing with significant scale or illumination changes in the scene.

- SIFT[12] (Scale-Invariant Feature Transform): SIFT is a widely recognized and highly robust algorithm known for its ability to detect and describe features that are invariant to scale, rotation, and illumination changes. Its robustness makes it well-suited for scenarios where the marker might be viewed from different angles or under varying lighting conditions. However, the computational complexity of SIFT can be a limiting factor in mobile applications where processing power is a constraint.

- SURF[14] (Speeded Up Robust Features): SURF is an alternative to SIFT that aims to achieve comparable performance with improved speed. It utilizes integral images and a Hessian matrix-based detector to identify interest points and constructs a descriptor based on Haar wavelet responses. SURF generally offers a better balance between speed and accuracy compared to SIFT, making it suitable for mobile AR applications where real-time performance is essential. However, its robustness to illumination changes might be less pronounced compared to SIFT.

- AKAZE[11] (Accelerated KAZE): AKAZE is a more recent feature detection and description algorithm that builds upon the KAZE algorithm. It focuses on non-linear scale space representation, improving the detection of features at varying scales. AKAZE has demonstrated superior performance compared to SIFT and SURF in terms of both speed and accuracy, making it a promising

candidate for mobile AR applications where both efficiency and robustness are critical.

It is important to note how all of the above algorithms are based on feature detection and description, which involves identifying distinctive points or regions in the image that can be matched across different frames to estimate the marker's pose. These methods are therefore image-processing-based and rely on the detection of keypoint features that are invariant to changes in scale, rotation, and illumination, thus can be considered traditional computer vision techniques.

More advanced machine learning-based approaches, such as Convolutional Neural Networks (CNNs) and deep learning architectures, have also shown promise in image recognition and pose estimation tasks. However, these methods often require significant computational resources and training data, and might also need either a complete retraining or fine-tuning of the model for each new marker or environment, which can be impractical for a user-generated content AR application like ShARED. The limitations and trade-offs of using machine learning-based approaches in marker-based AR applications will be discussed in the dedicated sections of this chapter.

Another characteristic of the aforementioned traditional computer based approach implemented in the ShARED app prototype is their reliance on monocular vision rather than stereo vision. Monocular vision refers to the use of a single camera to capture images and estimate the marker's pose, while stereo vision involves using two or more cameras to create a 3D representation of the scene. Stereo vision can provide more accurate depth information and improve the robustness of pose estimation, especially in scenarios with complex lighting conditions or occlusions. However, stereo vision requires additional hardware, computational resources and dedicated exposed APIs in the target devices' operating system, which might not be available for a mobile AR application like ShARED. A justification of the choice of monocular vision for the ShARED app will be provided in the dedicated section of this chapter.

3.2.4 Image Tracking

The Image Tracking module is responsible for maintaining the accurate and stable spatial registration of detected markers over time. This module ensures that the augmented content associated with a marker

remains anchored to the marker even as the user moves the device or the marker itself changes position. This involves continuously updating the marker's pose as new camera frames are captured and compensating for any potential drift or jitter in the tracking process.

Implemented Image Tracking Algorithms and Techniques

ShARED explores and implements a variety of image tracking techniques to evaluate their effectiveness and trade-offs in terms of accuracy, computational cost, and reliance on sensor data in the context of a mobile, marker-based AR application:

- AR Foundation/ARCore Native Tracking: The application leverages the native tracking capabilities provided by AR Foundation, which internally relies on platform-specific AR frameworks like ARCore on Android devices. These native tracking implementations are typically optimized for the target platform and potentially combine multiple sensor inputs, including the camera, IMU, and other available sensors (e.g. depth sensors). The specific algorithms employed are often proprietary and might involve sophisticated techniques like Simultaneous Localization and Mapping (SLAM) [1] and Visual Inertial Odometry (VIO) [2]. While these native tracking solutions offer high performance and robustness, they might be less flexible or customizable compared to custom tracking algorithms.
- Image Re-detection Tracking: This approach involves re-running the pose detection module for each incoming camera frame. The pose of the marker is recalculated from scratch, essentially treating each frame as a new detection instance. While straightforward to implement, this method can be computationally intensive, especially when using complex image recognition algorithms like SIFT or SURF. It might also lead to less smooth tracking performance compared to techniques that integrate temporal information or utilize sensor fusion.

[1] SLAM is a technique used to simultaneously build a map of an unknown environment and localize the device within that environment. In the context of AR, SLAM can be used to track the device's pose and create a 3D map of the surroundings, allowing for the placement of virtual objects that persist in the environment even when the device moves out of view and returns.

[2] VIO combines visual information from the camera with inertial measurements from the IMU (accelerometer and gyroscope) algorithms to estimate the device's motion. By fusing these two sources of information, VIO can provide more accurate and robust tracking, especially in environments with limited visual features or during rapid movements.

- Sensor-Based Tracking: This technique relies solely on the device's IMU data (accelerometer and gyroscope) to estimate the marker's pose changes over time. It assumes that the marker remains stationary relative to the environment, and any changes in the device's orientation or position are directly translated to the marker's pose. This method is computationally efficient and can provide reasonable tracking for short durations or when the device's movement is minimal. However, it is susceptible to drift over time, as errors in the IMU measurements accumulate, leading to inaccurate marker pose estimates and misalignment of augmented content.
- Sensor Fusion Tracking: This approach combines the strengths of both image-based and sensor-based tracking. It utilizes the image re-detection method to periodically obtain a more precise pose estimate for the marker and then uses IMU data to track the marker's pose between these updates. This fusion of sensor data can significantly improve tracking accuracy and reduce drift compared to using sensors alone.
  - Simple Sensor Fusion: A simple sensor fusion approach involves linearly interpolating the marker's pose between the image-based pose updates using the IMU measurements.
  - Kalman Filter: A more sophisticated sensor fusion technique is the Kalman Filter. A Kalman Filter is a recursive algorithm that estimates the state of a system based on a series of noisy measurements. It can be used to fuse data from multiple sensors, such as the camera and IMU, to obtain a more accurate and reliable estimate of the marker's pose. The Kalman Filter takes into account the uncertainties associated with both the sensor measurements and the system's dynamics, providing a statistically optimal estimate of the marker's pose over time. It is particularly effective in reducing the effects of sensor noise and drift, resulting in smoother and more accurate tracking performance.

Similar considerations for the use of machine learning or stereo vision techniques in image recognition and pose detection apply to image tracking as well. The current implementation of ShARED focuses on traditional computer vision and sensor fusion techniques for image tracking, prioritizing efficiency, accuracy, and responsiveness on mobile devices. The reasons for these choices and their trade-offs and implications will be discussed in the dedicated section of this chapter.

3.2.5 AR Objects Manager

The AR Objects Manager module is the heart of ShARED's content presentation and interaction capabilities. It handles the instantiation, positioning, and manipulation of augmented reality objects within the user's view. This module acts as a bridge between the tracked markers and the digital assets that constitute the augmented experience. Its core functions include:

- Object Instantiation and Association: When a marker is successfully detected and tracked, the AR Objects Manager retrieves the associated augmented objects from the experience data. These objects can be of various types, including 3D models, texts, images or videos. Each object is instantiated as a virtual entity within the AR scene and linked to its corresponding marker. The manager maintains this association, ensuring that the object's position and orientation are updated based on the marker's tracking information.
- Transform Management: The AR Objects Manager continuously updates the spatial transformations (position, rotation, and scale) of each AR object based on its associated marker's pose. This ensures that the object remains anchored to the marker and moves realistically as the user interacts with the physical environment. The manager also handles any default or user-defined offsets and transformations specified in the experience design, allowing for fine-grained control over object placement and alignment.
- Object State Management: Beyond spatial transformations, the manager also handles the state of each AR object. This state can encompass various attributes, including visibility, animation playback status, or any other properties that define the object's appearance and behavior. Together with the Multi-user Communication Module, the AR Objects Manager updates and synchronizes these states, ensuring that all connected users perceive the objects consistently.

3.2.6 User Interactions

The User Interactions module facilitates user engagement and control within the augmented reality environment. It captures and processes user input, translating it into meaningful actions that manipulate the AR objects or trigger specific events within the experience. This mod-

ule is crucial for creating interactive and engaging AR applications, enabling users to explore, learn, and play within the augmented world.

- Input Handling: The module captures various forms of user input (taps, drags, pinches, etc.), processing them to recognizing gestures and determining touch targets.
- Object Manipulation: Based on recognized user inputs, the module triggers actions on the AR objects. These actions can include:
  - Translation: Moving an object along a specific axis or plane.
  - Rotation: Rotating an object around an axis.
  - Scaling: Changing the size of an object uniformly or along specific dimensions.
- Event Triggering: User interactions can also trigger specific events or actions within the AR experience, like animation playback events (starting, stopping, or controlling the playback of videos or animations on 3D models) and information display events (activating pop-up windows, tooltips, or other UI elements that provide additional information or context related to the interacted object).

The User Interactions module is tightly coupled with the AR Objects Manager, working together to create ensure that user inputs are translated into meaningful actions that manipulate the augmented world.

3.2.7 Serialization/Deserialization Modules

The Serialization/Deserialization modules are essential for enabling the storage, sharing, and retrieval of AR environments created within ShARED. These modules handle the conversion of complex AR environment data into a compact and portable format that can be saved locally, uploaded to a remote server, and downloaded by other users.

Serialization

The Serialization module takes the in-memory representation of an AR environment, including marker data, associated object properties, spatial transformations, and any other relevant information, and converts it into a stream of bytes. This process involves:

- Data Transformation: Converting data structures, objects, and attributes into a format suitable for serialization. This might involve flattening hierarchical structures, converting object references to unique identifiers, and encoding data types appropriately.
- Data Encoding: Choosing an efficient and portable data encoding scheme, such as JSON, XML, or a custom binary format, to represent the serialized data.
- Compression (Optional): Applying compression algorithms to reduce the size of the serialized data, minimizing storage requirements and network transmission time.

Deserialization

The Deserialization module performs the reverse operation, taking a serialized byte stream and reconstructing the original AR environment in memory. This involves:

- Data Decoding: Parsing the serialized data stream according to the chosen encoding scheme, extracting individual data elements and their associated types.
- Data Reconstruction: Rebuilding the original data structures and objects from the decoded data elements, restoring object references, and populating attributes.
- Decompression (Optional): If compression was applied during serialization, the deserialization process must include a decompression step to restore the data to its original form.

3.2.8 Serialization Challenges

The main challenge for the serialization of AR environments for the ShARED application was the heterogeneity and complexity of the data involved. AR environments consist of a wide variety of data types, including images, 3D models, metadata, and spatial transformations, each requiring a different serialization approach. The serialization process needed to be flexible and extensible to accommodate these diverse data types while ensuring efficient storage and transmission of the serialized data.

In order to serialize the different data types involved in AR environments, the serialization module needs to support:

- Image Serialization: Converting image data (e.g., marker images, images to display as augmented objects, and textures for 3D models) into a format that preserves image quality and allows for efficient storage and transmission.
- Video Serialization: Encoding video data (e.g., animations, video textures) in a way that maintains video quality and playback performance while minimizing file size.
- 3D Model Serialization: Encoding 3D model data (e.g., mesh geometry, materials, animations) in a way that captures the model's structure and appearance while minimizing file size.
- Metadata Serialization: Storing metadata associated with AR objects, markers, and the overall environment in a structured format that can be easily reconstructed during deserialization.

The serialization module also needed to handle the serialization of spatial transformations, such as object positions, rotations, and scales, in a way that preserved their accuracy and precision. This required careful consideration of coordinate systems, units of measurement, and numerical representations to ensure that spatial transformations were faithfully reconstructed during deserialization.

Serialization Format and Compression Techniques

The choice of serialization format and the use of compression techniques directly impact the efficiency and performance of ShARED. A compact and efficiently serialized representation minimizes storage space on user devices and reduces network bandwidth consumption during sharing and downloading experiences. Additionally, the serialization/deserialization process should be robust, handling potential data inconsistencies or version mismatches between the sender and receiver.

Because the serialized AR environments can be complex and contain various data types, including images (in various formats, such as PNG or JPG/JPEG), 3D models (in a GLB format, the binary representation of the GLTF format for 3D models, which stands for "GL Transmission Format"), and metadata (in native file types such as text or binary), the serialization format must support a wide range of data structures and be easily extensible. JSON (JavaScript Object Notation) is a popular choice for its human-readable structure, wide compatibility with programming languages and platforms, and support for complex data

types. JSON provides a lightweight and flexible representation of AR environment data, making it suitable for storing and sharing experiences across different devices and environments. The use of JSON also simplifies the integration of the serialization/deserialization modules with the app's backend server, enabling seamless data exchange between the client and server components.

The prototype implementation of ShARED utilizes JSON as the primary serialization format, which provides a lightweight and flexible representation of AR environment data, making it suitable for storing and sharing experiences across different devices and environments. The use of JSON also simplifies the integration of the serialization/deserialization modules with the app's Python-based backend server, enabling seamless data exchange between the client and server components.

In the JSON serialized data of each AR experience, the main components are:

- Metadata Section: Contains information about the AR environment, such as the author's name, creation date, and a brief description. This metadata provides context and attribution for the AR experience, helping users understand the content and its origin.
- AR Objects: An array of AR objects present in the environment, each represented by a nested JSON object containing properties like the object's type (3D model, image, video, text), position, rotation, scale, metadata, and the associated augmented content, in turn serialized as:
  - Text Objects: Represented by a JSON object containing the simple text content to display, supporting HTML and Markdown formatting for rich text display.
  - Images: The image data (for both markers and image AR objects) is serialized as a Base64-encoded string, preserving the image's pixel information and format. This encoding ensures that the image can be reconstructed accurately during deserialization, maintaining its visual quality and integrity.
  - Videos: Video data is serialized similarly to images, using Base64 string encoding to represent the video file's binary content. This encoding preserves the video's frames and audio information, allowing for faithful reconstruction during deserialization.

– 3D Models: The only accepted upload format for 3D models in the prototype implementation of the app is the GLB format, which is a binary representation of the GLTF format for 3D models. The GLB binary data is encoded as a Base64 string and embedded directly within the JSON structure, ensuring a self-contained representation of the 3D model data.

For compression, the prototype implementation of ShARED does not apply compression to the serialized data stream, as the size of the serialized AR environments is manageable without compression. However, the app's architecture is designed to support future integration of compression algorithms, such as Gzip [3] or LZ4 [4], to reduce the size of serialized data further. This flexibility ensures that ShARED can adapt to varying storage and network requirements, optimizing performance and user experience based on specific deployment scenarios.

3.2.9 Local Storage, Remote Storage and Server Modules

The Local and Remote Storage modules are integral to the functionality of ShARED, enabling the persistence, sharing, and accessibility of AR experiences. These modules provide the mechanisms for saving and retrieving serialized AR environment data, both on the user's device and on a centralized server.

Local Storage

The Local Storage module manages the storage of AR experiences on the user's device. This allows users to create and save experiences directly on their smartphones or tablets, making them available for offline use.

The module provides functions for: saving new experiences (serializing and storing the AR environment data to the chosen local storage format, either by creating a new AR experience, updating an existing one, or downloading an online experience shared by another user from

[3] Gzip is a popular file compression format that reduces the size of files by compressing them using the DEFLATE algorithm. Gzip is widely supported and can significantly reduce the size of serialized data, making it faster to transfer over networks and requiring less storage space on devices.

[4] LZ4 is a fast compression algorithm that provides high compression and decompression speeds, making it suitable for real-time applications and scenarios where low latency is critical. LZ4 can be used to compress serialized data before storage or transmission, reducing the data size without introducing significant processing overhead.

the server), loading existing experiences (retrieving and deserializing stored experience data to reconstruct the AR environment in memory), and finally uploading or deleting experiences.

Remote Storage and Server

The Remote Storage module, paired with the Server module, extends ShARED's functionality by enabling the sharing and distribution of AR experiences among users. This cloud-based storage solution allows users to upload their experiences to a central server, making them accessible to a wider audience. The centralized Server component manages the storage, retrieval, and synchronization of AR experiences across multiple devices, ensuring consistency and availability of shared content.

The functionalities of the remote server include:

- Server Communication: Establishing and maintaining a connection between users and the remote server, handling data upload and download requests, and managing serialized AR experience data as well as user account information.
- User Authentication and Authorization: Implementing user accounts and authentication mechanisms to manage user-created experiences, control access to shared experiences, and potentially implement privacy settings.
- Experience Upload and Download:
  - Uploading: Serializing and transmitting user-created experiences to the server, potentially including metadata such as the experience title, description, author, and associated tags.
  - Downloading: Retrieving serialized experiences from the server based on user requests, including searching, browsing, and accessing shared experiences.
- Experience Management: The server-side component of this module is responsible for managing stored experiences, including:
  - Organizing experiences: Categorizing, tagging, and indexing experiences to facilitate discovery and browsing.
  - Handling versioning: Managing updates to existing experiences and ensuring compatibility between different versions of the ShARED app.

– Synchronizing experiences: Ensuring that shared experiences are consistent across all users and devices, updating them in real-time as changes are made.

Remote Server Implementation

The server-side and remote storage functionalities of the ShARED prototype app are implemented using a cloud-based solution, leveraging a RESTful API for communication between the client and server components.

The server for the prototype implementation of ShARED is hosted on a "Platform as a Service" (PaaS) [5] provider (PythonAnywhere [6]) which offers free hosting for small-scale applications as well as Python-based web services. The server is implemented using the Flask [7] framework, a lightweight and flexible Python web framework that provides the necessary tools for building RESTful APIs and web services, based on exposing simple web routes for client HTTP requests implemented as simple Python functions. The server's functionality is organized into different routes, each corresponding to a specific API endpoint that handles a particular type of request from the client app, such as uploading an AR experience, downloading a shared experience, or updating user account information.

The authentication of users is managed using a simple OTP (One-Time Password) exchange mechanism, where users receive a unique code via email upon registration or login, which they must enter in the app to verify their identity. This approach provides a basic level of security and ensures that only authorized users can upload and update their AR experiences on the platform, while also allowing for a streamlined and user-friendly authentication process which does not require the creation and management of passwords.

The server's data storage is implemented using a simple JSON database approach, where serialized AR experiences are stored as JSON files

[5] Platform as a Service (PaaS) is a cloud computing model that provides a platform and environment for developers to build, deploy, and manage applications without the complexity of infrastructure management. PaaS solutions typically include development tools, runtime environments, and scalable resources for hosting applications.

[6] PythonAnywhere is a cloud-based platform for hosting and running Python applications, providing a scalable and cost-effective solution for deploying web services and APIs.

[7] Flask is a lightweight and flexible Python web framework that provides the necessary tools for building RESTful APIs and web services, based on exposing simple web routes for HTTP requests implemented as simple Python functions.

on the server's Linux-based file system of PythonAnywhere. Each experience is associated with a unique identifier, allowing for easy retrieval and management of shared content. The server also maintains user account information, including usernames, email addresses, and authentication tokens, to facilitate user management and access control.

3.2.10 Multi-User Communication Module

The Multi-user Communication module is the responsible for the app's live sharing functionality, enabling multiple users to participate in and interact with the same AR environment simultaneously. This module leverages networking technologies to facilitate real-time communication and data synchronization between devices, creating a shared experience where users can see and manipulate augmented content together.

Networking Technologies

Various networking technologies were considered to establish communication channels between devices for live sharing in the app:

- Bluetooth: Bluetooth is a short-range wireless technology suitable for connecting devices within a limited proximity, typically up to 10 meters. It offers a relatively low bandwidth but can be energy-efficient, making it potentially suitable for small-scale, localized AR experiences.
- Wi-Fi Direct: Wi-Fi Direct enables devices to establish direct Wi-Fi connections without the need for a central access point. It provides higher bandwidth and range compared to Bluetooth, facilitating more data-intensive AR experiences and supporting a larger number of connected devices.
- Local Area Network (LAN): A LAN connects devices within a limited area, such as a classroom or home network, through a wired or wireless connection. LANs typically offer high bandwidth and low latency, making them well-suited for robust multi-user AR interactions.
- Remote Server: Using a remote server to relay communication between devices introduces significant latency due to data transmission over the internet. This approach is generally less suitable

for real-time, interactive AR experiences, particularly those relying on precise synchronization and low-latency feedback.

Netcode for GameObjects

ShARED utilizes the Unity's "Netcode for GameObjects" library to simplify the development of multi-user AR experiences. This library provides a high-level networking framework that abstracts the complexities of low-level network communication, allowing to focus on implementing game logic and data synchronization during development.

Key features of "Netcode for GameObjects" include:

- Client-Server Architecture: Netcode typically employs a client-server architecture, where one device acts as the server, managing the shared AR environment and synchronizing data between connected clients.
- Object Synchronization: Netcode automatically handles the synchronization of designated game objects across the network. This includes synchronizing object transformations (position, rotation, scale) and other relevant properties, ensuring that all users see a consistent representation of the shared environment.
- Network Events and Messages: Netcode provides mechanisms for sending custom events and messages between clients and the server, allowing developers to implement specific game logic and interactions within the shared environment.
- Network Prediction and Interpolation: Netcode can implement network prediction and interpolation techniques to compensate for network latency, creating a smoother and more responsive experience for users despite potential network delays.

Implementation of Multi-user Communication

ShARED leverages the capabilities of the "Netcode for GameObjects" library to establish communication between devices and synchronize AR content for live sharing:

- Connection Establishment: The app allows users to create or join live sharing sessions. Depending on the chosen networking technology, devices connect directly using Bluetooth or Wi-Fi Direct,

or through a LAN connection (e.g. the same Wi-Fi network). One device is designated as the server, which acts as a central node for managing connections and relaying data between other users, as clients.

- Data Synchronization: The server maintains an authoritative representation of the shared AR environment, including the current state of all synchronized objects. When a client interacts with an object (e.g. moves, rotates, or changes its state), the client sends an update to the server. The server processes the update, applies it to the authoritative environment, and then broadcasts the updated state to all connected clients.
- Handling Network Events: NetCode's network event system is used to handle specific interactions or events within the shared environment, such as triggering animations, playing audio, or displaying notifications to all participants.

Shared AR Experience State Representation

The state of the shared AR experience is represented as a simple collection of AR objects associated to each image marker: the one-to-one mapping between markers and AR objects makes the synchronization process straightforward. Each AR object's state can be simply defined by its position, rotation, scale, and any additional properties or metadata (e.g. playback time for videos or animated 3D models). No spatial information needs to be shared between devices, as the AR objects' positions are determined locally by the markers' poses, which are tracked independently on each device and which are always associated to a single AR object. This simplifies the synchronization process and reduces the amount of data that needs to be transmitted between devices, improving the efficiency and responsiveness of the shared experience.

Networking Technologies Considerations

The choice of networking technology influences the range, bandwidth, and latency of the live sharing experience.

For small-scale, localized experiences, Bluetooth might suffice, while Wi-Fi Direct or LAN connections are better suited for larger groups or more data-intensive interactions.

The use of a remote server for live sharing is generally discouraged due to the increased latency and potential network congestion, which can degrade the user experience and hinder real-time interactions. Furthermore, the close spatial proximity required for multi-user AR experiences makes LAN or Wi-Fi Direct connections more suitable for maintaining low-latency communication between devices, and thus renders the use of a remote server as the centralized host for the live-sharing of AR experiences unwise and impractical.

The developed prototype of ShARED utilizes a LAN connection for live sharing, leveraging the high bandwidth and low latency of local networks to ensure a responsive and interactive multi-user experience: the Netcode for GameObjects library simplifies the implementation of networked interactions for real-time synchronization of AR content across connected devices on the same Wi-Fi network.

3.3 Marker Detection, Pose Estimation, and Tracking

This section discusses the methodologies implemented in the ShARED prototype for marker detection, pose estimation, and tracking, which form the core of the application's augmented reality capabilities. The section provides an in-depth explanation of the algorithms, implementation details, and challenges associated with each functionality.

While the focus here is on comprehensively describing the approaches used and their integration within the application, the performance analysis and experimental results of these methodologies are presented in Chapter 4.

3.3.1 Marker Detection

Marker detection is a fundamental step in marker-based augmented reality, as it allows the application to recognize user-defined planar images and use them as anchors for augmented content. This process involves identifying and matching features in a marker image with the corresponding features in a frame captured by the device's camera.

Feature Detection Algorithms

The ShARED prototype implements marker detection using well-established feature-based algorithms, specifically ORB, SIFT, SURF, and the more recent AKAZE. Each of these algorithms provide different properties and advantages, and follow a similar two-step process for image recognition: feature extraction first, then feature matching [7].

Feature extraction involves identifying key points in an image, such as corners or edges, that are invariant to changes in scale, rotation, and lighting. These key points are then described using numerical descriptors that capture their unique properties. Feature matching compares the descriptors extracted from the marker image to those extracted from the current camera frame to identify corresponding key points. Once sufficient matches are found, the marker image can be considered as detected in the frame.

Each of the algorithms used offers distinct advantages and trade-offs:

- ORB (Oriented FAST and Rotated BRIEF): A computationally efficient algorithm that combines the FAST key point detector and the BRIEF descriptor, optimized for real-time applications.
- SIFT (Scale-Invariant Feature Transform): A robust algorithm that provides high accuracy and reliability in feature detection and matching, particularly under varying lighting and perspective conditions, but with higher computational requirements.
- SURF (Speeded-Up Robust Features): A faster alternative to SIFT with slightly reduced accuracy, suitable for mobile platforms with limited processing power.
- AKAZE (Accelerated KAZE): An efficient algorithm designed for nonlinear scale-space representations, offering a balance between speed and accuracy.

Implementation in ShARED

The ShARED application uses the OpenCV library, via its C# Unity wrapper (OpenCVForUnity), for feature extraction and matching. The implementation consists of two phases: pre-processing and runtime detection.

During pre-processing, each user-defined marker image is analyzed to extract its feature descriptors, which are stored in the feature descriptor library associated with the AR experience. This step is performed immediately after the user captures, crops and adjusts the marker image during the editing process. Cropping ensures that only the relevant portions of the image are used for detection, improving both accuracy and computational efficiency.

At runtime, the application uses the feature descriptors to match key points between the marker images and each frame captured by the device's camera. Once matches are identified, a homography matrix is computed using the matched points, allowing the marker's position and orientation relative to the camera to be determined for pose estimation (described in the next section).

The use of OpenCV ensures that the feature detection and matching processes are efficient and robust, enabling the application to perform marker detection in real time while maintaining compatibility with the limited computational resources of mobile devices.

3.3.2 Pose Estimation

Pose estimation refers to the process of determining the position and orientation of a detected marker in the real-world coordinate system relative to the camera. This functionality is critical for overlaying augmented content correctly within the physical environment.

Homography-Based Methods

Various pose estimation methods based on both traditional computer vision techniques and machine learning exist[3]. In the ShARED prototype, pose estimation was implemented using homography-based methods. Since the application relies on planar 2D markers, the relationship between the marker's image coordinates and its real-world coordinates can be represented by a homography matrix.

The homography matrix is computed from the matched feature points between the detected marker and the corresponding reference image. Using these points, the matrix provides a transformation that maps points in the marker image to points in the camera frame. This transformation encapsulates the marker's position and orientation relative to the camera, assuming that the scale of the marker is provided during

its creation.

Unlike methods based on 3D point matching, such as Perspective-n-Point (PnP) [8] algorithms, which are more complex and computationally intensive, homography-based approaches are simpler and computationally efficient for planar markers. They also avoid the need for additional camera calibration data, as the necessary parameters are handled internally by the Unity and OpenCV frameworks.

Implementation in ShARED

In the ShARED prototype, pose estimation is tightly integrated with the marker detection process. Once a marker is detected in a camera frame, the matched feature points are used to compute the homography matrix with OpenCV's functionality. The application then uses the homography and the known physical scale of the marker to determine its pose relative to the camera.

The computed pose is subsequently passed to the AR Objects Manager module, which is responsible for rendering the augmented content associated with the marker. This module uses the marker's pose to position, orient, and scale the augmented content accurately within the virtual 3D space overlaid onto the physical environment.

3.3.3 Marker Tracking

Marker tracking involves continuously determining the position and orientation of a detected marker across successive frames. This functionality ensures that the augmented content associated with the marker remains accurately aligned with the real world, even as the camera or marker moves. Various tracking methods exist, based on image-processing techniques, machine learning and/or sensor measurements(1), each with its own trade-offs in terms of accuracy, computational efficiency, and responsiveness: the ShARED prototype integrates multiple simplified tracking approaches to study their performance and suitability for mobile AR applications.

[8] Perspective-n-Point (PnP) is a pose estimation technique that uses the correspondence between 3D points in the real world and their 2D projections in the camera image to estimate the camera's pose. PnP methods require knowledge of the 3D points' coordinates and their corresponding 2D image points, which can be challenging to obtain for arbitrary objects.

Image-Based Tracking

In the ShARED prototype, one method of marker tracking relies on image-based techniques, where the position and orientation of the marker are re-estimated for each camera frame. This process uses the same marker detection and pose estimation pipeline described in the previous sections but omits the initial step of feature extraction for the reference marker, as its descriptors are already precomputed.

While this approach provides accurate tracking as long as the marker remains in the camera's field of view, it is computationally expensive when performed at every frame. To optimize performance, the application includes strategies to distribute detection and pose updates over multiple frames, which are discussed in a later section.

Sensor-Based Tracking

The second method of tracking relies on the device's inertial measurement unit (IMU), which provides data from accelerometer and gyroscope sensors. These sensors measure the device's motion and orientation in real time. Using this data, the application estimates the device's pose changes relative to the previous frame, allowing it to approximate the marker's updated position and orientation.

While sensor-based tracking is computationally efficient and does not require continuous visual detection of the marker, it introduces drift errors over time due to the inherent inaccuracies of IMU measurements. These errors are especially pronounced during fast or abrupt device movements, which are common when using handheld smartphones.

Sensor Fusion for Tracking

To overcome the limitations of relying solely on image-based or sensor-based tracking, the ShARED prototype integrates the two methods using sensor fusion techniques. This approach combines the high accuracy of image-based tracking with the responsiveness of sensor-based tracking, leveraging their complementary strengths.

Two sensor fusion methods are implemented in the prototype:

- Linear Interpolation: This simple method combines the outputs of the two tracking approaches by interpolating between the image-

based pose estimate and the sensor-based pose estimate, weighted by their respective confidence levels.

- Kalman Filter fusion: This probabilistic approach estimates the marker's pose by combining measurements from both sources, accounting for their uncertainties. The Kalman Filter smooths the tracking output and reduces the impact of IMU drift errors, particularly during rapid camera movements or temporary loss of visual detection.

By employing these fusion techniques, the ShARED application achieves robust and reliable marker tracking, ensuring a seamless augmented reality experience even under challenging conditions, such as fast motion or partial occlusion of the marker.

3.3.4 Performance Optimizations

Performance optimization is crucial for ensuring that the ShARED prototype delivers a responsive and real-time augmented reality experience, especially given the computational limitations of mobile devices.

Frame Processing Strategies

To optimize the marker detection and tracking process, various techniques were explored to reduce the computational load associated with processing every camera frame. These include:

- Frame-by-Frame Detection: The simplest method processes all markers in the AR environment's library for each captured frame. While this approach ensures high detection accuracy, it imposes significant computational overhead, making it unsuitable for real-time applications.
- Per-Marker Cycling: This method processes only one marker per captured camera frame, cycling through all markers over successive frames. It reduces computational demand while ensuring all markers are periodically checked, but at the cost of potentially slower updates for individual markers.
- Detect Every α: A fixed detection rate, denoted as α, is used to limit full marker detection to every $1/\alpha$ frames. This balances accuracy and performance by distributing the app's image processing workload throughout frames.

- Distributed Detection: This approach combines a detection rate parameter with distributed processing, dividing markers across multiple frames. For example, with N markers and a detection rate $1/kN$, all markers are processed only once every k frames, ensuring an even distribution of computation.
- On-Demand Detection: Detection is triggered manually by the user, reducing unnecessary processing during periods when markers are unlikely to change. This method is particularly useful in scenarios where users already know the location and timing of marker interactions.

Figure 3.2 provides a visual representation of these frame processing strategies, illustrating how they distribute marker detection and tracking across multiple frames to balance performance and accuracy.

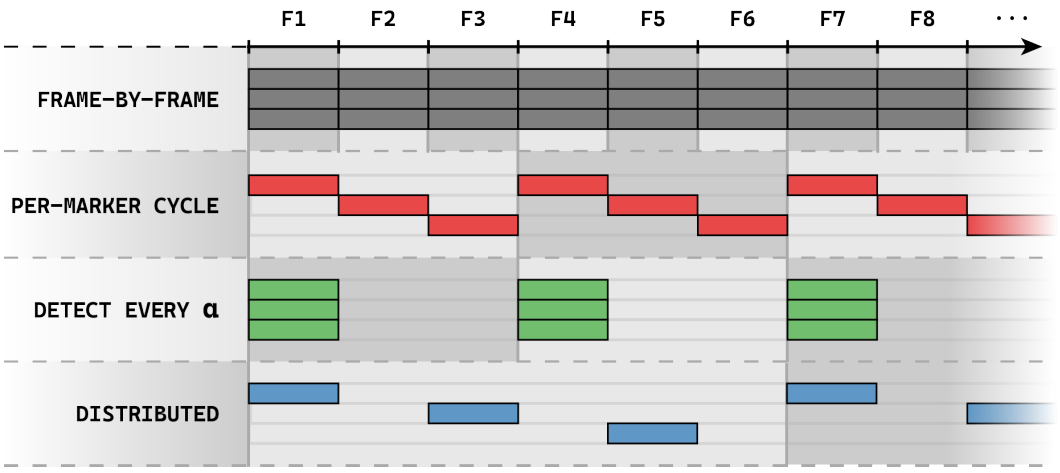


Figure 3.2 Frame Processing Strategies for Marker Detection and Tracking, showing N=3 frames for all methods, α=1/3 for "Detect Every α" and k=2 for "Distributed Detection".

Another optimization strategy employed in the ShARED prototype involves leveraging the device's accelerometer sensor data to dynamically control the activation and deactivation of computationally intensive components, such as the image recognition and pose estimation modules. By analyzing the accelerometer readings, the application can detect periods of significant motion, which often result in blurry camera frames. During such instances, attempting image recognition or pose estimation would be computationally wasteful and potentially less accurate. Therefore, the application pauses these processes when the accelerometer detects excessive movement, effectively reducing computational load and conserving battery life.

Trade-Offs in Performance

Each strategy involves trade-offs between real-time responsiveness, detection accuracy, and computational efficiency. For instance, distributed detection and on-demand methods prioritize resource savings, while frame-by-frame detection ensures maximum accuracy.

Testing results for each of these strategies are provided in chapter 4, where their impact on the frame rate of the application will be evaluated in detail. Initial implementation results indicate that distributed detection offers a practical compromise, particularly for experiences involving multiple markers, while on-demand detection is suitable for scenarios where the user already knows which real-world objects could trigger AR content in the environment.

3.3.5 Challenges and Final Considerations

Developing the ShARED prototype required addressing several challenges and evaluating critical considerations related to marker detection, pose estimation, and tracking. These challenges stem from hardware limitations, algorithmic trade-offs, and the inherent constraints of mobile augmented reality applications.

Monocular vs. Stereo Vision

The ShARED application uses monocular vision, relying on a single camera to perform marker detection and pose estimation. While this approach is sufficient for planar 2D markers, it limits depth estimation accuracy compared to stereo vision systems, which utilize multiple cameras to achieve higher precision.

Modern mobile devices often include multiple cameras capable of stereo vision. However, the ability to access multi-camera APIs is not universally supported. For example, although Android introduced multi-camera API support with version 9 (API level 28), its availability depends on the device manufacturer. In the case of the HUAWEI Mate 20 Lite, the test device used for the ShARED prototype, multi-camera API access was not exposed despite the presence of dual rear cameras. This constraint necessitated reliance on monocular methods for marker pose estimation to ensure compatibility with most modern devices.

Stereo vision could significantly enhance the accuracy of marker tracking and pose estimation by providing additional depth information. However, its integration into the ShARED application remains limited by hardware accessibility and platform compatibility.

Machine Learning-Based Approaches

Recent advancements in machine learning, particularly deep learning, have improved the accuracy and robustness of marker detection and tracking. However, integrating these methods into a mobile application like ShARED presents significant challenges:

- Computational Overhead: Machine learning (ML) models, especially deep neural networks, require substantial computational resources for inference, which can exceed the capabilities of mobile devices.
- Real-Time Constraints: The dynamic nature of the ShARED app requires immediate feedback during marker creation and AR interactions, making it impractical to both perform inference on-device or offload ML-centered computations to remote servers due to potential latency issues.
- Dynamic Marker Creation: ShARED supports user-defined markers created in real-time, precluding the use of pre-trained models designed for static datasets. On-device training or model updates would introduce additional performance and usability challenges.
- Resource Efficiency: Mobile devices have limited memory and battery life, and the computational demands of machine learning models can impact the overall user experience.

Some of the challenges presented above may be partially mitigated by leveraging lightweight ML models designed for mobile platforms or employing cloud-based ML services. In the first case, lightweight ML models offer a balance between accuracy and efficiency, making them suitable for real-time applications like ShARED, but trade-offs between performance, accuracy, and resource consumption would still be necessary. Cloud-based ML services, on the other hand, can offload computation to remote servers, reducing the burden on mobile devices, but introduce latency and connectivity issues that may affect the app's responsiveness, and may also introduce additional requirements, such as higher bandwidth and data usage, as well as raised privacy concerns due to potentially sensitive data transmission to external servers.

Although AR SDKs such as ARCore and ARKit likely incorporate proprietary machine learning methods optimized for mobile devices, their implementations remain inaccessible for customization. Furthermore, their implementation in native OS libraries can benefit from hardware-level optimizations which would be inaccessible to third party applications. For these reasons, the ShARED prototype prioritizes traditional computer vision techniques, which provide an adequate balance of speed, accuracy, and resource efficiency.

Platform and Hardware Limitations

The capabilities of mobile AR applications are influenced by the underlying hardware and platform-specific features. In addition to the previously mentioned multi-camera API limitations, other constraints may include:

- Sensor Accuracy: IMU sensors on mobile devices often introduce drift errors, particularly during rapid movements or rotations.
- Lighting Conditions: Marker detection can be affected by low light or uneven illumination, reducing the reliability of feature matching.
- Device-Specific Optimizations: AR libraries like ARCore and ARKit leverage proprietary optimizations for hardware-level integration, which may vary across devices and platforms.
- Performance Variability: Mobile devices exhibit performance variations based on factors such as battery level, temperature, and background processes, affecting the app's real-time responsiveness.
- OS and API Compatibility: The availability of features like multi-camera APIs, sensor fusion libraries, and low-level camera access can vary between different Android and iOS versions, impacting the app's functionality.

These challenges highlight the importance of designing adaptable and efficient algorithms that account for the variability in hardware and software environments, and may make it harder to achieve better performances and trade-offs for marker detection, pose estimation, and tracking when using custom-made algorithms and components with respect to the ones provided by the native AR SDKs (ARCore and ARKit). Despite these limitations, custom solutions still offer a higher

degree of flexibility and control over the AR experience, allowing for tailored optimizations and feature enhancements that may not be possible with off-the-shelf solutions (like prioritizing detection accuracy over speed by employing a specific detection algorithm).

Chapter 4

Experimental Evaluation and Analysis

This chapter presents the experimental evaluation of the ShARED prototype, focusing on the performance of the implemented image recognition, pose detection, and image tracking components and their related optimization methods.

The chapter is structured into several sections. First, the experimental setup is detailed, including the testing environment, device specifications, and the tools used for ground truth generation and data analysis. Next, the results of the image recognition and pose detection tests are presented, followed by the evaluation of the image tracking methods. A separate section discusses the impact of the optimization strategies on the application's frame rate and performance. The chapter concludes with a discussion of the findings, highlighting trade-offs and implications for the design and scalability of the ShARED application.

4.1 Experimental Setup

This section describes the setup used to conduct the experimental evaluation of the ShARED prototype. It includes the details of the hardware and software environment, the tools and methods used for generating ground truth data, and the metrics employed to evaluate the performance of the implemented algorithms and optimizations.

4.1.1 Testing Device and Environment

All experiments were conducted on a HUAWEI Mate 20 Lite smartphone, running Android 10. The device is equipped with a 1080x2340

resolution display at 409 dpi, 4 GB of RAM, and an octa-core CPU with two 2.2 GHz cores and a third 1.7 GHz core. This hardware configuration represents a typical mid-range mobile device, suitable for evaluating the performance of the ShARED application on consumer-grade smartphones.

The ShARED app was run directly on this device, with results logged during execution. Data such as frame rates, recognition times, and pose estimation results were collected for analysis using in-app logging mechanisms and Unity's built-in profiler tool.

4.1.2 Test Dataset

To evaluate the accuracy of image recognition, pose detection, and tracking algorithms, test data was collected and manually annotated using a custom-made web tool.

The tests used a dataset of 50 images grouped into 5 scenes, with each scene containing 10 consecutive frames. To each of the 5 scenes, a different marker image was assigned, with associated metadata for the width and height of the marker image. The test dataset was specifically selected to cover a range of scenarios, including variations in lighting conditions, marker orientations, and camera movements, to evaluate the robustness and reliability of the algorithms under different conditions. Figures 4.1 and 4.2 show the marker images and camera frames used for testing.

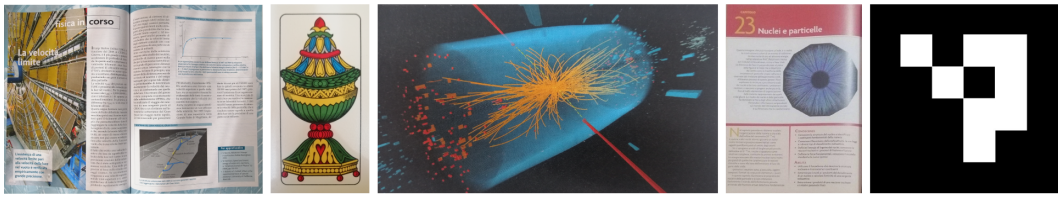
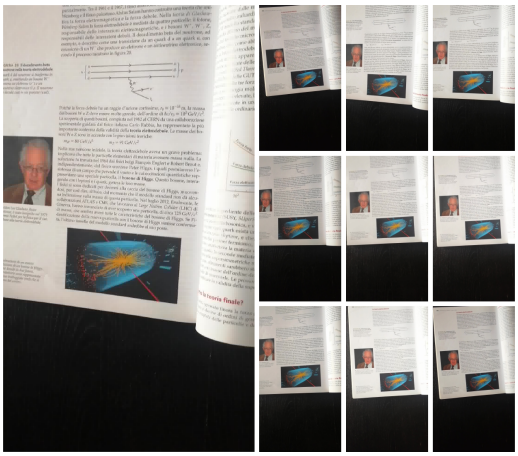


Figure 4.1 Markers to detect for each test camera frames.

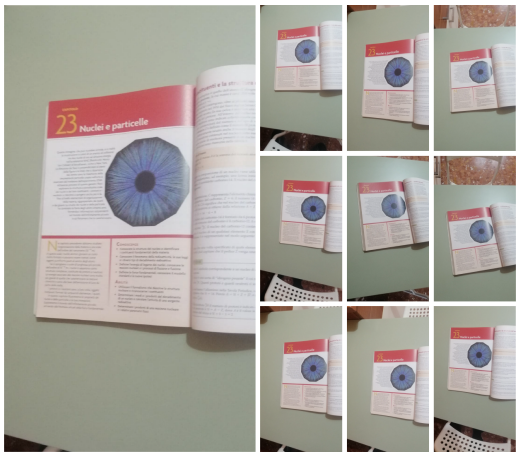
The ground truth position and rotation of the images in the test frames were established using a custom-developed web tool (built using HTML and JavaScript). This tool enabled precise manual annotation by allowing the user to define the positions of the four corners of each marker (top-left, top-right, bottom-left, and bottom-right) within the frame. Using the known real-world dimensions of the marker and these annotated corners, it was then possible to calculate the marker's



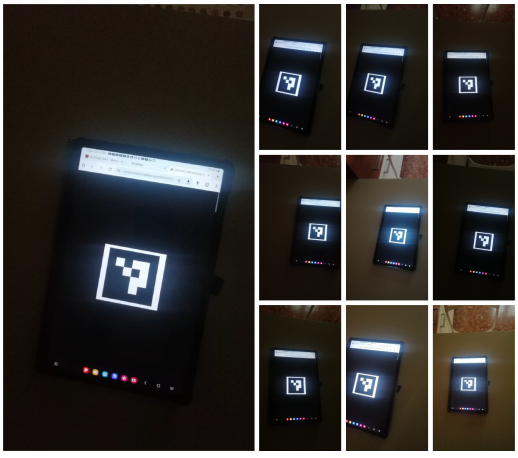
Scene #1 Frames

Scene #2 Frames

Scene #3 Frames



Scene #4 Frames



Scene #5 Frames

Figure 4.2 Camera frame images used for testing.

position and orientation in the camera's coordinate system for each frame, used as ground truth data to evaluate the accuracy of the image recognition, pose detection, and tracking algorithms.

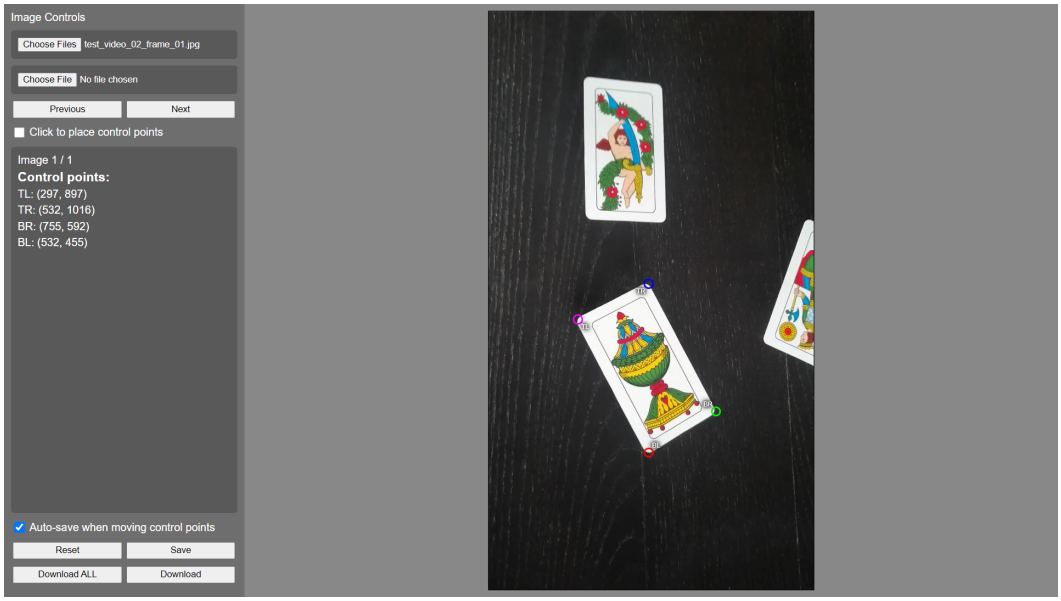


Figure 4.3 Ground truth sample data labeling tool.

4.1.3 Evaluation Metrics

Three key metrics were used to assess the performance of the implemented components:

- **Accuracy**: For image recognition and pose detection, this metric measures the percentage of correctly recognized markers and the error in position and rotation of the estimated pose compared to the ground truth. For tracking, it evaluates the accuracy of pose updates over successive frames.
- **Speed**: This metric measures the time required for image recognition, pose detection, and tracking in milliseconds, excluding any pre-processing steps such as feature extraction.
- **Performance**: This metric evaluates the real-time behavior of the application by measuring the frames per second (FPS) achieved during various operations and optimization configurations.

The experimental results were processed using a Python-based tool developed to calculate errors, averages, and comparative metrics for

the tested algorithms and optimization strategies. Graphs and charts summarizing the results, generated once again using a custom-made Python tool, are included in the following sections of this chapter to provide a clear visualization of the findings.

4.2 Image Recognition and Pose Detection Evaluation

This section presents the results of the experimental evaluation of the image recognition and pose detection components of the ShARED app, measuring the accuracy and speed of the related implemented methods. The tested algorithms include ORB, SIFT, SURF, AKAZE, and the native AR Foundation image recognition and pose detection algorithms provided by ARCore.

4.2.1 Accuracy

The accuracy of the image recognition and pose detection algorithms was evaluated by calculating:

- The percentage of correctly recognized markers across the test camera frames.
- The error in the estimated position and rotation of the detected marker pose compared to the ground truth data.

Each of the test camera frames were processed independently by each of the algorithms, and the results were compared to the ground truth data to calculate the recognition rate and pose estimation errors.

A summary of the accuracy results for each algorithm is provided in Table 4.1, showing the percentage of correctly recognized markers and the average position and rotation errors for each algorithm.

Figures 4.4, 4.5, and 4.6 show scatterplot graphs for the position error (X axis) and rotation errors (Y axis) of the tested algorithms (only showing points corresponding to correctly recognized frames). In particular, Figure 4.4 shows position and rotation errors for all camera frames of all of the tested algorithms, Figure 4.5 shows position and rotation errors for each tested algorithm, and Figure 4.6 shows position and rotation errors for each sample scene.

Algorithm	Recognition Rate (%)	Position Error (cm)	Rotation Error (degrees)
ORB	54	0.932	2.2
SIFT	94	0.205	1.2
SURF	80	0.487	1.6
AKAZE	16	0.357	1.3
AR Foundation	90	0.231	0.8

Table 4.1 Accuracy test results for Image Recognition and Pose Detection algorithms

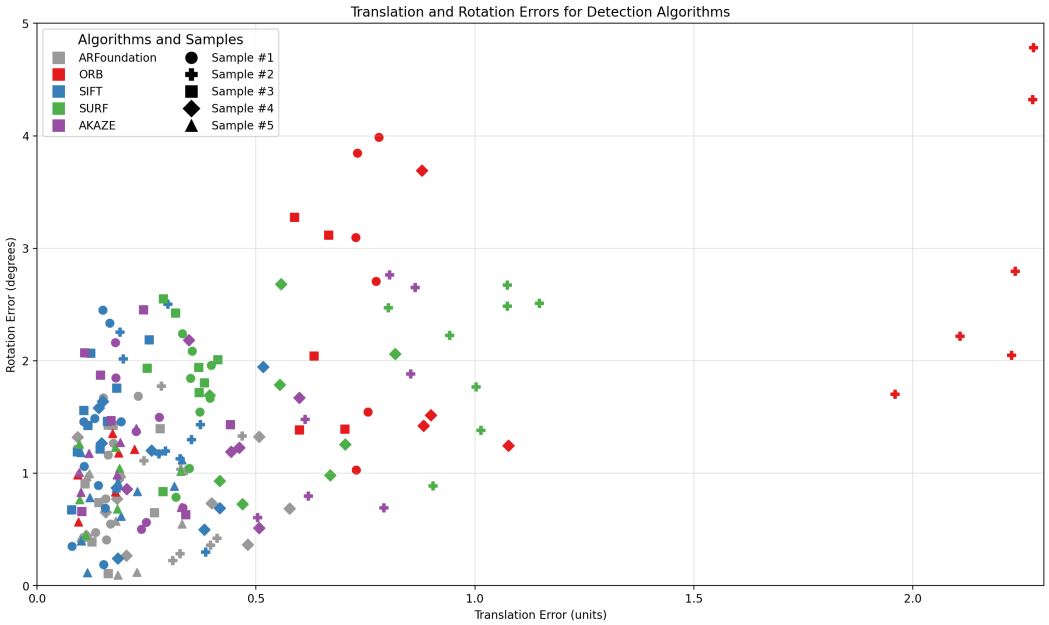
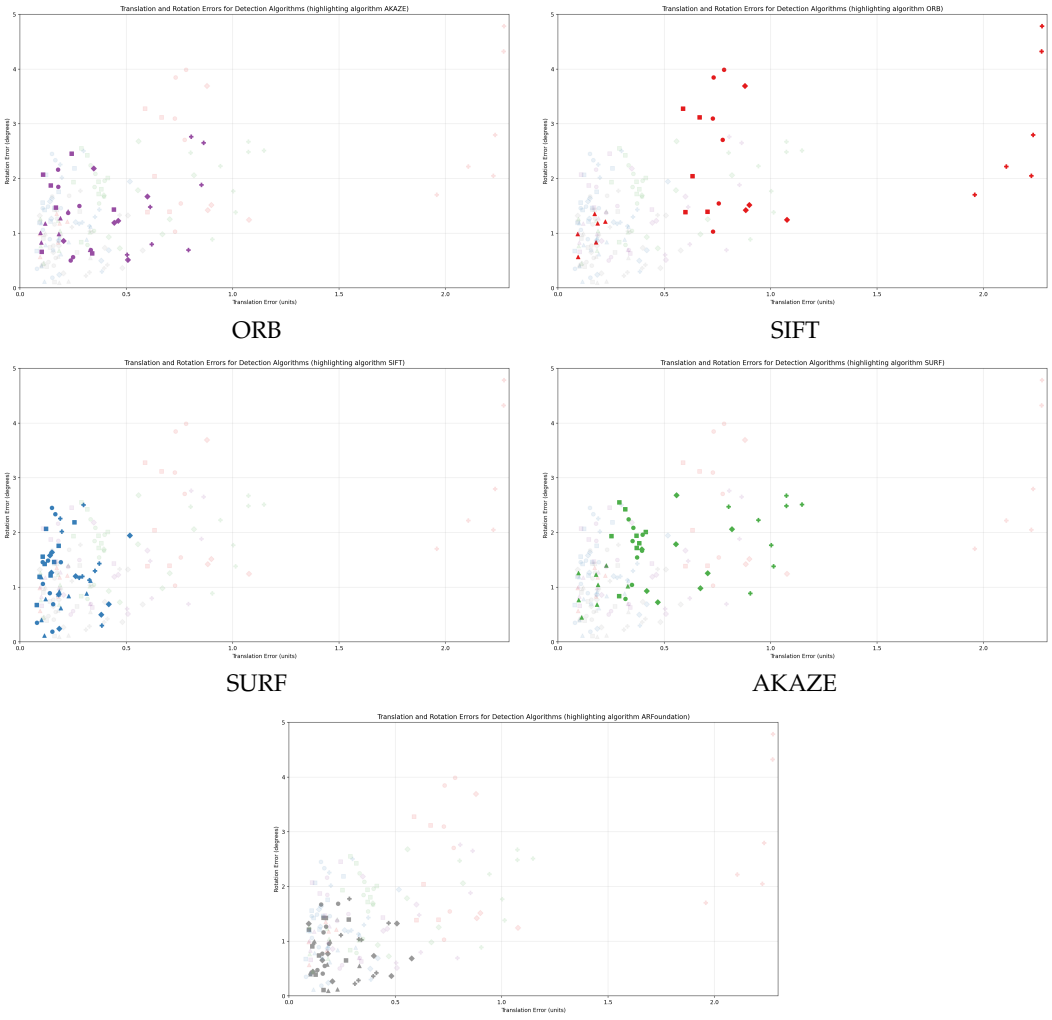


Figure 4.4 Detection position and rotation errors.

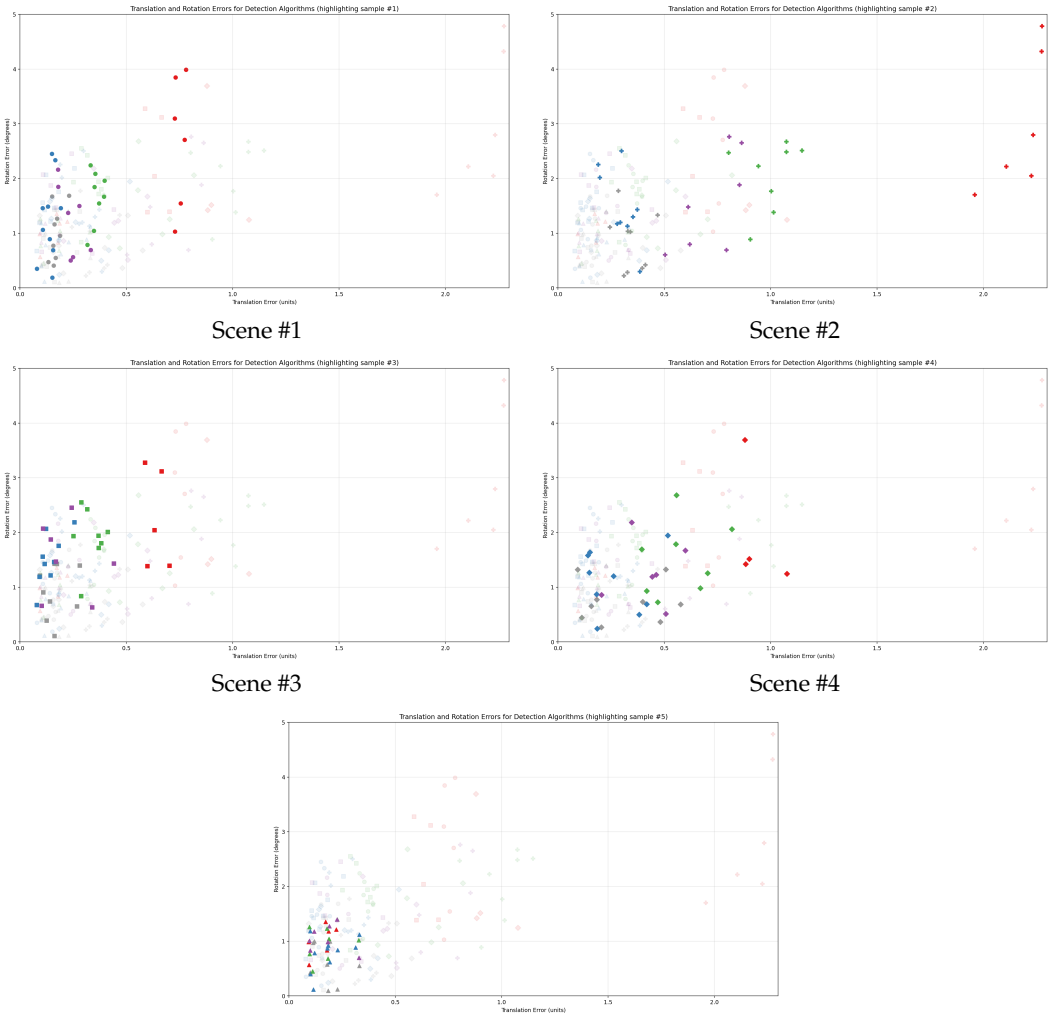


ORB SIFT

SURF AKAZE

AR Foundation

Figure 4.5 Detection position and rotation errors for each tested algorithm.



Scene #1 Scene #2

Scene #3 Scene #4

Scene #5

Figure 4.6 Detection position and rotation errors for each sample scene.

4.2.2 Speed

The speed of image recognition and pose detection was measured as the time required to identify a marker in a frame and estimate its pose, excluding pre-processing steps for computing the feature descriptors of each marker image. Measurements were collected in milliseconds and averaged over multiple runs for each algorithm.

Table 4.2 summarizes the speed results for each tested algorithm, showing the average time taken to recognize a marker and estimate its pose (along with a total time taken for both phases), and also providing an estimate of the frames per second (FPS) achievable by each algorithm given its measured total execution time.

Algorithm	Recognition Time (ms)	Detection Time (ms)	Total Time (ms)	Frame Rate (FPS)
ORB	12	16	28	35
SIFT	68	54	122	8
SURF	32	32	64	15
AKAZE	27	14	41	24
AR Foundation	13	6	19	52

Table 4.2 Speed test results for Image Recognition and Pose Detection algorithms

4.2.3 Discussion of Results

The experimental evaluation of the image recognition and pose detection algorithms in the ShARED prototype provided valuable insights into the accuracy and speed of the implemented methods. The results showed that the SIFT algorithm achieved the highest recognition rate and very low position and rotation errors, while the AR Foundation algorithm provided the fastest recognition and pose estimation times while still retaining a high performance on the pose detection task: this higher overall performances of the AR Foundation library, and thus the native AR Core Android library, can once again be attributed to the use of optimized algorithms, machine learning and deep learning methods, and hardware acceleration provided by the ARCore platform, as discussed in Section 3.3.5. The AKAZE algorithm, while presenting a lower recognition rate, showed a good balance between

accuracy and speed, making it a viable alternative to the other tested algorithms. The ORB algorithm, despite having a lower recognition rate, performed very well in terms of speed, making it suitable for real-time applications where speed is a priority: furthermore, it showed a faster image recognition speed than the AR Foundation library's image recognition module, making the ORB detector a good candidate for "Full Screen" AR experiences (discussed in Section **??**) which don't require marker tracking. The SURF algorithm, while providing a relatively good accuracy, was slower than the other algorithms, making it less suitable for real-time applications.

A comparison on the performance of the tested algorithms in terms of accuracy and speed is provided in Figure 4.7, showing a parallel coordinates plot of the recognition rate, position error, rotation error, and total execution time for each algorithm.

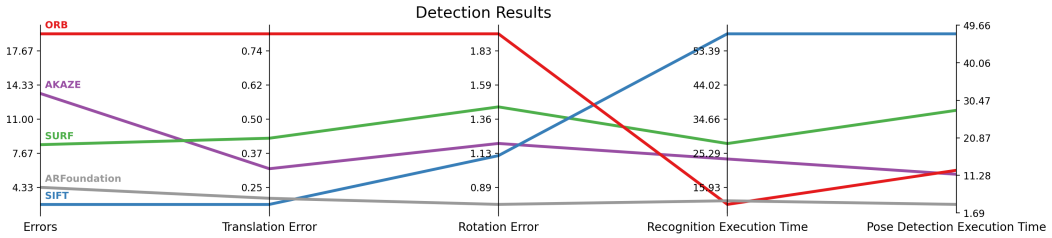


Figure 4.7 Performance comparison of image recognition and pose detection algorithms.

4.3 Image Tracking Evaluation

This section presents the results of the experimental evaluation of the image tracking methods implemented in the ShARED prototype. The tested methods include image re-detection tracking, sensor-only tracking, simple and Kalman-based fusion (of image re-detection and sensor-based tracking). Tests focused on measuring tracking accuracy and speed, providing a comparative analysis of their effectiveness and performance.

4.3.1 Accuracy

The accuracy of image tracking was evaluated using 45 pairs of consecutive frames extracted from the 50 test frames described in section 4.1.2, with each pair treated independently. For each frame pair, the ground truth position and rotation of the marker in both frames were

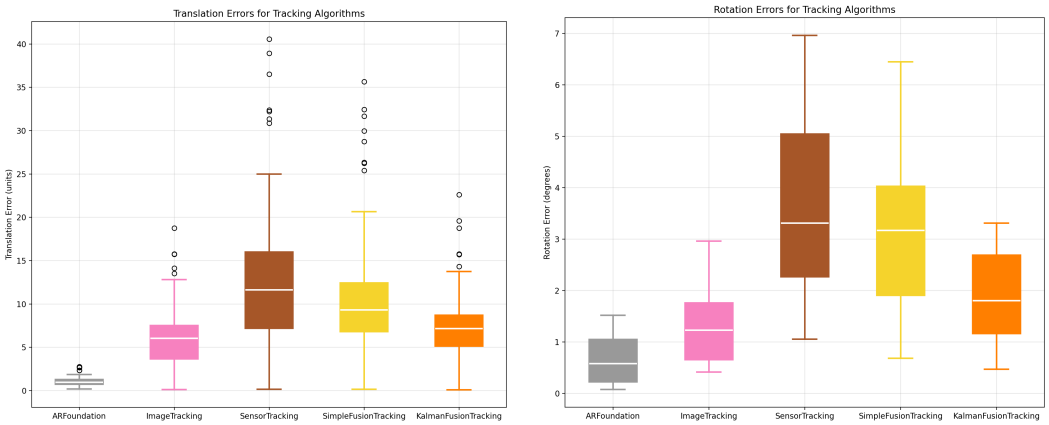
compared with the estimated pose generated by the tracking algorithms to calculate the position and rotation errors.

Table 4.3 summarizes the accuracy results for each tested tracking method, showing the average position and rotation errors for each algorithm.

Algorithm	Position Error (cm)	Rotation Error (degrees)
Image Tracking	13.1	3.5
Sensor Tracking	7.1	1.4
Simple Fusion	11.9	2.9
Kalman Fusion	7.7	2.0
AR Foundation	1.1	0.7

Table 4.3 Accuracy test results for Image Tracking algorithms

Figure 4.8 shows boxplot graphs for the position and rotation errors of the tested tracking algorithms, providing a comparative analysis of the accuracy of each method.



Tracking Position Error Tracking Rotation Error

Figure 4.8 Tracking position and rotation errors for each tested algorithm.

4.3.2 Speed

The speed of each tracking method was measured as the time required to compute the updated pose for a marker in consecutive frames.

Table 4.4 summarizes the speed results for each tested tracking method, showing the average time taken to compute the updated pose of a marker from the tracking phase data, in milliseconds, and the estimated frames per second (FPS) achievable by each algorithm given its measured execution time.

Algorithm	Execution Time (ms)	Frame Rate (FPS)
Image Tracking	22	45
Sensor Tracking	3	333
Simple Fusion	24	41
Kalman Fusion	29	34
AR Foundation	9	111

Table 4.4 Speed test results for Image Tracking algorithms

4.3.3 Discussion of Results

The experimental evaluation of the image tracking methods in the ShARED prototype provided valuable insights into the accuracy and speed of the implemented algorithms. The results showed that, while the AR Foundation algorithm achieved the highest accuracy and very low position and rotation errors, the sensor-only tracking method provided the fastest tracking times, making it suitable for real-time applications where speed is a priority: it is also worth noting that sensor-based tracking methods present an increasing error in both position and rotation as frames progress because of sensor drift, for which errors accumulate over each new frame (thus a refresh of the real anchor position should be triggered periodically). The simple fusion tracking method, while showing a higher position and rotation error than the sensor-only tracking method, provided a good balance between accuracy and speed, making it a viable alternative for applications where a higher accuracy is required. The Kalman-based fusion tracking method, while showing a lower position and rotation error than the simple fusion tracking method, was slower, making it less suitable for real-time applications.

A comparison on the performance of the tested tracking algorithms in terms of accuracy and speed is provided in Figure 4.9, showing a

parallel coordinates plot of the position error, rotation error, and total execution time for each algorithm.

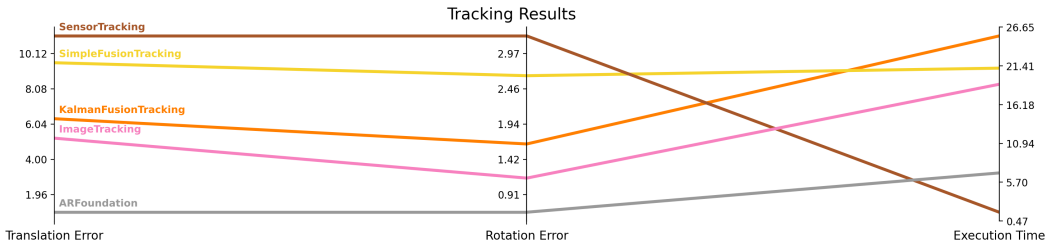


Figure 4.9 Performance comparison of image tracking algorithms.

4.4 Optimization Testing and Evaluation

This section evaluates the performance impact of the optimization methods described in Section 3.3.4. The tests measured the actual frame rate (frames per second, FPS) of the ShARED prototype under different optimization strategies.

4.4.1 Performance of Optimization Methods

The following four optimization methods were tested (see Section 3.3.4 for more details on each method):

- Frame-by-Frame Detection: Processing all markers in each frame.
- Per-Marker Cycling: Cycling through markers frame-by-frame.
- Detection Rate Parameter: Processing all markers every M frames.
- Distributed Detection: Processing one marker at a rate of $1/kN$, cycling through N markers every kN frames.

The underlying image recognition and pose detection algorithm used for each of the tests of these optimization methods were the native AR Foundation library algorithms.

Each method was evaluated by running the application for a total of 200 frames on the test device with 5 markers in the markers library, and FPS was logged using both Unity's profiler and in-app logging mechanisms.
For per-marker cycling, we used $M = 10$ (for a final detection rate parameter equal to $\alpha = 0.1$), while for distributed detection we set

$k = 2$. The results were averaged over 5 runs to provide a reliable estimate of the performance impact of each optimization strategy.

Table 4.5 summarizes the performance results for each optimization method, showing the average frames per second (FPS) achieved by the ShARED prototype under different optimization configurations.

Optimization Method	Avg. Frame Rate (FPS)
Frame-by-Frame Detection	18.7
Per-Marker Cycling	35.8
Detect Every α	43.4
Distributed Detection	47.5

Table 4.5 Performance Results for Optimization Methods

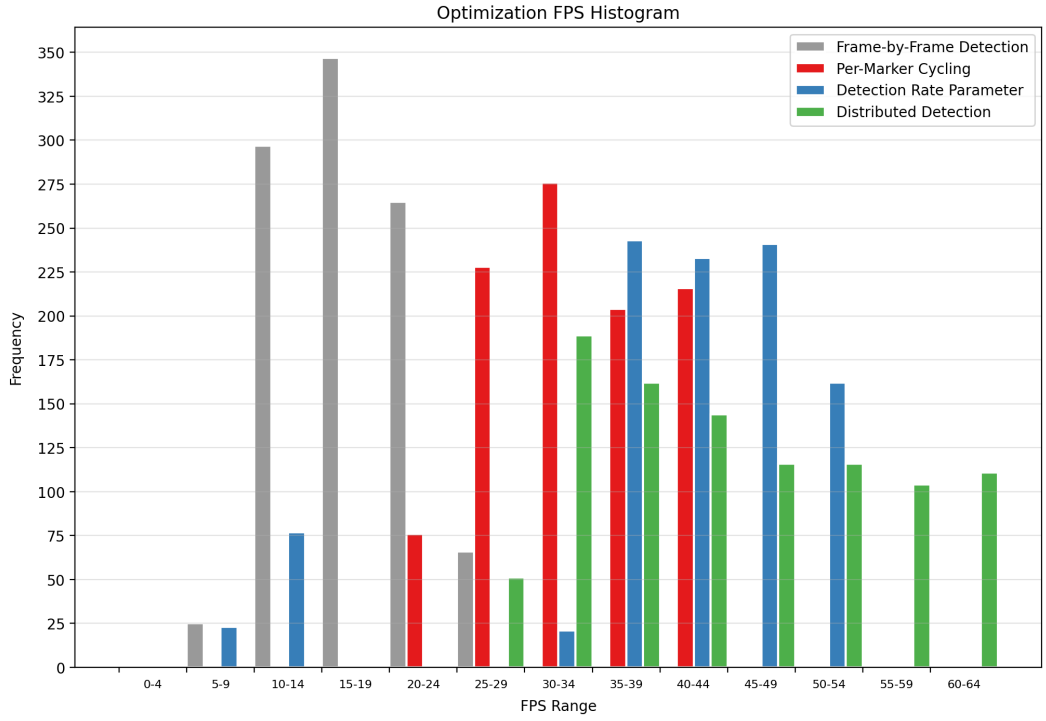


Figure 4.10 FPS comparison of optimization methods.

4.4.2 Discussion of Results

The experimental evaluation of the optimization methods in the ShARED prototype provided valuable insights into the performance impact of each strategy. The results showed that the distributed detection method achieved the highest frames per second (FPS) rate, followed by the detection rate parameter method, per-marker cycling, and frame-by-frame detection. The distributed detection method, which processes one marker at a rate of $1/kN$, cycling through N markers every kN frames, for sufficiently small values of k, provides a good balance between accuracy and speed, making it the most suitable optimization strategy for the ShARED application. The detection rate parameter method, which processes all markers every M frames, achieved a similarly good frame rate, while the per-marker cycling method, which cycles through markers frame-by-frame, showed only a moderate improvement in FPS over the non-optimized frame-by-frame detection method, which processes all markers in each frame: the latter, while possibly providing the best results in terms of responsiveness of the application to new marker images entering the camera frame, achieved the lowest FPS rate, making it the least suitable optimization strategy for real-time applications.

Chapter 5

Discussion, Use Cases and Future Directions

This chapter discusses the concepts and findings of the ShARED thesis project, focusing on the main application idea presented in Chapter 2, the development process, challenges, and considerations outlined in Chapter 3, as well as the experimental results presented in Chapter 4.

The chapter also explores potential use cases for the ShARED app, highlighting its applications in education, entertainment, and other domains, including both general ideas and specific scenarios where the application could be beneficial and impactful.

Finally, future directions for the ShARED project are outlined, focusing on enhancements, extensions, and research opportunities for the development of the application.

5.1 Challenges and Discussion

The development and evaluation of ShARED revealed both the potential and limitations of marker-based augmented reality (AR) on mobile devices. The project aimed to create a flexible, open-source framework for shared AR experiences, enabling users to create and share their own AR content and environments: the development process involved designing and implementing a cohesive and comprehensive modular application by implementing custom computer vision algorithms, integrating AR Foundation libraries, and optimizing performance for mobile devices. The following sections discuss the encountered challenges, the implications of the experimental results, and the potential trade-offs and considerations for the design and scalability of the

ShARED application.

5.1.1 Challenges

While developing the ShARED prototype, several challenges and potential limitations of marker-based AR on mobile devices were identified, most notably:

- Limitations of Marker-Based AR: Marker-based AR relies on physical markers, such as images or patterns, to anchor augmented content, which introduce certain constraints:
  - Users must have access to the physical or displayed markers, which can be inconvenient in some scenarios.
  - Environmental factors, such as inadequate lighting, occlusion, or excessive camera movement, can hinder marker detection and tracking.
  - The number of markers that can be detected simultaneously is limited, restricting the complexity and interactivity of AR experiences.
  - The accuracy and robustness of marker detection and tracking depend on the quality of the markers and the performance of the image recognition algorithms.
  - Marker-based AR experiences may lack context awareness and adaptability to dynamic environments.
- Computational Constraints on Mobile Devices: Smartphones, while powerful, are inherently limited compared to desktop systems, and running computationally intensive algorithms such as those for real-time image recognition and pose estimation may impact performance. This trade-off between accuracy, processing speed, and battery consumption must be carefully managed to ensure a smooth and responsive user experience.
- Challenges in Tracking and Sensor Integration: AR systems often use feature detection techniques and hardware sensors for tracking, which face multiple challenges:
  - Feature detection-based tracking struggles on surfaces with insufficient distinctive features, such as planes with no visible edges or corners.
  - Sensor-based tracking, reliant on devices like accelerometers and gyroscopes, suffers from error accumulation over time

(sensor drift), leading to significant positional errors, especially during large or abrupt movements.
  - Sensor fusion methods, such as Kalman filters, require careful tuning and calibration to balance accuracy and responsiveness and may introduce additional computational overhead.
  - The integration of tracking methods with image recognition and pose estimation algorithms can be complex and may require custom solutions to optimize performance, which may clash with the use of general-purpose AR libraries or with the modular design of the application, relying on independent components for each task.

- Performance Optimization and Scalability: The real-time nature of AR applications necessitates efficient algorithms and optimization strategies to maintain responsiveness and interactivity. However, optimizing performance without sacrificing accuracy or usability is a delicate balance:
  - Frame-by-frame detection, while ensuring real-time responsiveness, may lead to high computational load and low frame rates, affecting user experience.
  - Optimization methods, such as per-marker cycling or distributed detection, must balance accuracy and speed, requiring careful parameter tuning and testing to achieve optimal performance.
  - The choice of image recognition and tracking algorithms, as well as the integration of sensor data, can impact the overall performance and scalability of the application, requiring trade-offs between accuracy, speed, and computational efficiency.
  - The scalability of the application, in terms of the number of markers, complexity of AR content, and interactivity of shared experiences, must be considered to ensure a seamless and engaging user experience.

5.1.2 Discussion

The ShARED project represents an innovative approach to shared augmented reality experiences, combining marker-based image recognition, pose detection, and tracking algorithms to create a real-time collaborative AR environment. The development process involved

designing and implementing custom computer vision algorithms, integrating AR Foundation libraries, and optimizing performance for mobile devices. The experimental evaluation of the ShARED prototype provided valuable insights into the accuracy, speed, and efficiency of the implemented components, highlighting the trade-offs between different algorithms and optimization strategies.

The landscape of augmented reality applications is rapidly evolving in recent years, with new technologies, platforms, and use cases emerging across various domains, but despite these significant advancements, the development of AR experiences and environments remains a complex and challenging task, oftentimes requiring a deep understanding of computer vision, machine learning, and mobile development concepts, or the use of proprietary AR SDKs and libraries or third-party applications and software that can be limiting in terms of customization and flexibility. The ShARED project aimed to address these challenges by providing a flexible, open-source framework for shared AR experiences, enabling developers as well as educators, researchers, and creators without specialized coding, computer vision, editing software or augmented reality knowledge to create and share their own AR content and experiences.

The developed prototype demonstrated that a potential solution for bridging the gap between the complexity of AR development and the accessibility of AR experiences could be achieved by means of a widely available, easy-to-use, and customizable smartphone application that leverages the capabilities of modern mobile devices and computer vision algorithms as well as the power of shared experiences and collaborative environments. The ShARED app, while still in its early stages of development, showed to be a versatile and scalable platform for creating simple as well as advanced AR experiences, with potential applications in education, entertainment, and other domains.

The experimental evaluation of the ShARED prototype provided valuable insights into the performance of the implemented image recognition, pose detection, and tracking algorithms, highlighting the accuracy, speed, and efficiency of each method. The results showed that the built-in solution for AR capabilities in Android devices, provided by the AR Foundation library and ARCore platform, achieved the highest accuracy and speed, making it the most suitable choice for real-time AR applications. However, the custom computer vision algorithms implemented in the ShARED app, including ORB, SIFT, SURF,

and AKAZE, provided viable alternatives for scenarios where customizability, flexibility, and performance optimizations are required in specific use cases. The use of the mobile app as a gateway to visualize augmented content associated to real-world objects and images (that is, the development of "Full Screen" AR experiences) and the experimental results obtained from the optimization tests, showed that some of the tested algorithms, like the ORB detector, could provide better performances than the default AR Foundation library's image recognition module, making them suitable for specific use cases and scenarios: this also frees the application from the use of performance and memory heavy general libraries (like ARCore), allowing for a more lightweight and customizable solution. Furthermore, the optimization methods tested in the ShARED app, including frame-by-frame detection, per-marker cycling, detection rate parameter, and distributed detection, provided valuable insights into the performance impact of each strategy, highlighting the trade-offs between accuracy, speed, and computational performance for each optimization method: for specific experiences with a small number of markers to detect and track, the distributed detection method showed to be the most suitable optimization strategy, providing a good balance between accuracy and speed without compromising the overall performance of the application. The user-triggered marker detection, despite not being tested in the experimental evaluation, could also show to be useful in scenarios where the user knows when to trigger the detection of a specific marker, thus reducing the computational load on the device and improving the overall performance of the application, and possibly once again freeing the application from the use of the bloated ARCore library. The experimental evaluation also showed that the sensor-based tracking method provided the fastest tracking times, making it suitable for real-time applications where speed is a priority and where the movement of the device is limited, thus reducing the impact of sensor drift on the tracking accuracy.

The modular design of the ShARED app, with separate components for image recognition, pose detection, and tracking, allows for easy customization and extension of the application, enabling developers to add new features, algorithms, and optimizations to enhance the user experience and performance of the app. The open-source nature of the project, with code available on GitHub[(4)], provides a valuable resource for the AR community, enabling collaboration, contributions, and feedback from developers, researchers, and creators interested in shared AR experiences and collaborative environments.

5.2 Use Cases

The ShARED app offers a wide range of use cases and applications in various domains, including education, entertainment, gaming, professional fields, and everyday life. The following sections explore some of the potential use cases for the application, highlighting its versatility, flexibility, and scalability in creating shared AR experiences.

5.2.1 General Use Cases

Most of the use cases of the ShARED application revolve around using AR to visualize and access information and content that would otherwise be hard or impossible to access in a physical, 2-dimensional form, such as accessing videos associated to static images, visualizing complex 3D structures or models, or interacting with virtual objects and characters in a real-world environment.

A general overview of the potential use cases for the ShARED app can be summarized by the following broad categories of AR experiences and applications:

- Education: ShARED can be used as an educational tool to create interactive and collaborative learning experiences. Teachers and students can develop AR content to visualize complex concepts, historical events, scientific phenomena, or architectural structures, fostering engagement and understanding in the classroom. The shared AR environments can enable group projects, virtual field trips, and interactive quizzes, enhancing the learning experience and promoting collaboration among students.
- Entertainment: The app can be used to create interactive games, puzzles, and storytelling experiences that engage users in immersive and entertaining AR worlds. Players can explore virtual environments, solve challenges, interact with digital characters and objects, or even augment the content of existing physical board games and experiences, blurring the lines between the physical and digital realms. The live-sharing capabilities of the ShARED app can also enable multiplayer games, social interactions, and collaborative storytelling, providing a new dimension of fun and creativity.
- Social Interaction: ShARED can facilitate social interactions and communication by enabling users to share AR content, messages,

and experiences with friends, family, and colleagues. Users can create personalized AR messages, virtual gifts, or shared spaces to connect with others in a unique and engaging way. Collaborative AR environments can also be used for virtual meetings, remote collaboration, or social events, bridging the gap between physical and digital interactions.

- Professional Fields: The app can be applied in professional fields such as architecture, engineering, design, and marketing to visualize concepts, prototypes, and products in augmented reality. Professionals can create interactive 3D models, architectural visualizations, product demos, or marketing campaigns to showcase their work and engage clients or customers. Shared AR environments can facilitate collaboration, feedback, and decision-making processes, enhancing productivity and communication in professional settings.

- Collaborative Projects: The app can be used for collaborative projects, research, and creative endeavors that require shared AR environments. Teams of developers, designers, artists, or researchers can work together to create interactive prototypes, visualizations, or simulations in augmented reality. The live-sharing capabilities of the ShARED app can facilitate real-time collaboration, feedback, and iteration, enabling teams to co-create and interact with AR content in a seamless and engaging way.

- Interactive Exhibits and Installations: ShARED can be used to create interactive exhibits, installations, and experiences for museums, galleries, events, and public spaces. Artists, curators, and designers can develop immersive and engaging AR content to enhance visitor engagement, storytelling, and exploration. Shared AR environments can enable interactive tours, educational experiences, or artistic installations that blur the boundaries between physical and digital art, providing a new medium for creative expression and audience interaction.

- Remote Assistance and Training: The app can be used for remote assistance, training, and support in various fields, such as maintenance, repair, healthcare, or education. Experts, instructors, or mentors can guide users through complex tasks, procedures, or exercises by overlaying instructions, visual cues, or annotations in augmented reality. Shared AR environments can facilitate remote collaboration, learning, and problem-solving, enabling real-time interactions and feedback between users and experts.

- Augmented Events and Experiences: ShARED can be used to create augmented events, experiences, and installations for festivals, conferences, exhibitions, or public gatherings. Organizers, artists, and creators can develop interactive and immersive AR content to engage audiences, enhance storytelling, or provide contextual information. Shared AR environments can enable interactive exhibits, scavenger hunts, or virtual performances that transform physical spaces into dynamic and interactive AR worlds, creating memorable and impactful experiences for participants.
- Personalized Augmented Content: The app can be used to create personalized and customized AR content for individual users, such as virtual diaries, memory albums, or interactive stories. Users can augment their physical surroundings with digital notes, photos, videos, or animations to create personalized AR experiences that reflect their memories, emotions, or interests. Shared AR environments can enable users to share their stories, experiences, or creations with others, fostering connections, empathy, and creativity in a unique and meaningful way.

5.2.2 Specific Use Case Scenarios

This section presents specific use case scenarios of the ShARED app, as standalone AR experiences for some of the categories presented above, providing a more detailed insight into the potential applications and benefits of the application.

Augmenting Educational Content in Books

The ShARED app demonstrates its potential as a powerful educational tool by enhancing the traditionally static content of school or university textbooks. Using marker-based AR, users can interact with augmented 3D models, transforming the way complex concepts are visualized and understood. A notable example is the augmentation of textbooks in various possible fields to visualize complex 3D structures and concepts which are oftentimes hard to grasp in the 2-dimensional format proper of textbook images and diagrams:

- Mathematics: Visualize 3D mathematical functions, graphs, and geometric shapes to deepen understanding and intuition.
- Mechanical Engineering: Visualize 3D moving parts, mechanisms, or dynamic assemblies directly from diagrams in textbooks.

- Civil Engineering: Augment structural models, such as bridges, tunnels, or roads, and overlay geographical terrain maps for detailed understanding.
- Aerospace Engineering: Explore complex aircraft components, aerodynamic models, and simulation data interactively.
- Medicine: View 3D medical imaging, such as MRI or CT scans, allowing for layer-by-layer exploration of anatomical structures.
- Biology: Simulate molecular structures and interactions, visualizing protein folding or chemical bonding in 3D.
- Architecture: Examine architectural designs as fully interactive 3D structures to understand their spatial dynamics and construction details.
- Physics: Simulate 3D motion of objects, visualize electromagnetic fields, or illustrate space body interactions.

These use cases allow users to interact with 3D representations of abstract or complex concepts, allowing to rotate, zoom, and explore the models from different angles, providing a more intuitive, engaging and effective learning experience and allowing to better understand complex concepts that are proper of the fields mentioned above and that are hard to visualize in a 2D textbook format.

Augmenting Printed Photos and Personal Memories

ShARED provides a novel way to add interactive and multimedia layers to personal photographs, bringing printed memories to life. Users can associate augmented content such as text or videos with printed photos, offering a richer and more personalized experience.

For instance, a user could scan a family photo album, and the app would overlay a video of the moments leading up to that photo or a text annotation describing the context or emotions of the memory. This feature is particularly valuable for framed photos displayed at home or images in a photobook, to which animated versions of photos (like videos taken with a smartphone alongside photos in a family trip or an important moment) or text notes to describe the subject and situation of the photo could be associated. Unlike traditional solutions such as QR codes or handwritten notes, the augmented content is seamlessly integrated with the photo and can be privately shared with others via a secure code, making it accessible only to chosen recipients.

Enhancing Social Board Games and Card Games

ShARED offers a creative solution to enhance physical board games and card games with augmented reality, introducing interactive features that complement traditional gameplay. For instance, in a tabletop role-playing game (T-RPG), players could scan specific cards or items on the game board to reveal additional information, such as character stats, storyline prompts, or hidden clues. The app can also display an interactive game map, dynamically updating as the players progress through the game. In collaborative games, players can use ShARED to take shared notes, overlay strategy hints, or view animations triggered by specific game events. For physical card games, the app could display card abilities, rules clarifications, or real-time effects, making the game more engaging and accessible for new players.

5.3 Future Directions

ShARED offers a modular design that allows for several promising extensions to its capabilities. This modular design, combined with the open-source nature of the project[(4)], provides a solid foundation for future development and research in the field of shared augmented reality. The following outlines potential enhancements, features, and research directions for the ShARED application.

- Markerless AR Integration: Future iterations of ShARED could incorporate markerless AR techniques, such as Simultaneous Localization and Mapping (SLAM) and Visual Inertial Odometry (VIO) techniques, to enable content anchoring without physical markers. This enhancement would make the app more flexible and accessible, while also addressing the limitations of marker-based AR, such as the need for physical markers and environmental constraints, providing the users with more options and flexibility in creating and sharing AR content.
- Generative AI for Content Creation: Recent advances in AI, and most notably generative AI tools, have enabled the creation of realistic images, 3D models, and animations. The integration of generative AI tools could empower users to create AR content more efficiently. AI-driven features could include the generation of images, 3D models, or textual summaries, with human oversight ensuring the quality and relevance of the content and simultaneously overcoming the limitations of modern generative AI technologies (like inconsistencies and possibly noisy outputs).

- Advanced AR Object Types: The app could extend its support to include interactive AR elements, such as clickable buttons, dynamic graphs, 3D models with explorable sub-components or sound/audio content. These additions would enhance the utility of AR experiences in fields such as education, engineering and entertaining, which require interactive and engaging content for effective learning and visualization.
- Cloud Computing and Storage: To address mobile hardware constraints, ShARED could leverage cloud computing for heavy processing tasks and storage. This would enable more complex AR applications and seamless sharing of environments across devices, while also ensuring scalability and performance optimization for resource-intensive tasks.
- Support for Emerging AR Devices: ShARED could evolve to support AR-enabled smart glasses, such as AR glasses and VR visors with pass-through capabilities. This adaptation would pave the way for new use cases, including everyday augmented experiences and immersive applications in virtual or mixed reality.
- Geo-Based AR Experiences: Future versions of the app could utilize geographic data to trigger AR content, enabling context-aware applications in tourism, navigation, and localized education. This functionality could be integrated with marker-based or markerless AR to provide a hybrid experience that adapts to the user's location and surroundings.
- Machine Learning integration: The recent advancements in machine learning and computer vision could be leveraged to enhance the image recognition and tracking capabilities of ShARED, especially assuming a continuation of the upwards trend in the performance of these algorithms and the availability of more powerful hardware in mobile devices. The integration of machine learning models for image recognition, pose estimation, and tracking could enhance the accuracy and robustness of the AR algorithms in ShARED, as discussed in Section 3.3.5, opening up the possibility of more complex and interactive AR experiences.

Chapter 6

Conclusion

This thesis has presented the design, implementation, and evaluation of ShARED, a mobile application that democratizes the creation, sharing, and experiencing of marker-based augmented reality (AR) environments. The app addresses key challenges in AR accessibility by enabling users to create personalized AR experiences directly on their smartphones, bypassing the need for external software or specialized technical expertise.

ShARED stands out by providing a seamless, mobile-centric platform for designing marker-based AR content. The application's ability to utilize images as anchors for augmented content allows users to create precise and contextually rich AR environments. Furthermore, the integration of sharing and collaborative features ensures that AR experiences can reach broader audiences and facilitate real-time interactions among users.

The application was designed with a modular architecture to ensure flexibility and future extensibility, which also allowed to explore different techniques and possibilities for the implementation of the core components of the application, such as image recognition, pose estimation, and tracking algorithms. Through the development of ShARED, several technical challenges were addressed, including optimizing marker detection and tracking, ensuring robust performance on mobile hardware, and enabling the synchronization of shared experiences in local environments. The use of AR Foundation and its integration with underlying ARKit and ARCore technologies was pivotal in achieving a reliable and efficient implementation. Moreover, the app's user-friendly design framework ensures that AR creation is accessible to individuals from diverse technical backgrounds.

The evaluation of ShARED demonstrated its effectiveness in various use cases, including educational content augmentation, interactive storytelling, and collaborative learning environments. These applications underscore the potential of marker-based AR to enhance user engagement and foster creativity across multiple domains.

Despite its achievements, ShARED's current implementation has limitations that pave the way for future development. Enhancing the app with markerless AR capabilities could expand its applicability to scenarios where predefined markers are impractical. Additionally, integrating generative AI tools for content creation offers an exciting opportunity to simplify the design process further. This could include AI-powered image recognition, automated 3D model generation, and intelligent content suggestions, all supervised by the user to maintain quality and relevance.

Looking forward, the potential adaptation of ShARED to emerging AR hardware, such as smart glasses, and its translation into VR environments present promising avenues for exploration. These developments could further amplify the impact of ShARED, transforming it into a versatile tool for immersive experiences in both personal and professional contexts.

In conclusion, ShARED represents a significant step towards making AR technology more accessible, customizable, and collaborative. By lowering the barriers to entry, this application encourages a wider audience to engage with and contribute to the growing field of augmented reality. The insights and methodologies discussed in this thesis provide a foundation for future innovations that continue to bridge the gap between digital and physical worlds.

Bibliography

[1] Omar Abdelaziz, Mohamed Shehata, and Mohamed Mohamed. Beyond Traditional Single Object Tracking: A Survey, 2024.

[2] Nakul Chhabhaiya, Bhumeshwar Patle, and Praveen Bhojane. Virtual & Augmented Reality Applications: A Broader Perspective. *Journal of Data Science and Intelligent Systems*, 01 2024.

[3] Jian Guan, Yingming Hao, Qingxiao Wu, Sicong Li, and Yingjian Fang. A Survey of 6DoF Object Pose Estimation Methods for Different Application Scenarios. *Sensors*, 24(4), 2024.

[4] Valerio Di Stefano. Shared App GitHub repository. `https://github.com/valeriodiste/shared_app`, 2024. Licensed under the GNU General Public License v3.0.

[5] Philipp A. Rauschnabel, Reto Felix, Chris Hinsch, Hamza Shahab, and Florian Alt. What is XR? Towards a Framework for Augmented and Virtual Reality. *Computers in Human Behavior*, 133:107289, 2022.

[6] Pietro Cipresso, Irene Alice Chicchi Giglioli, Mariano Alcañiz Raya, and Giuseppe Riva. The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature. *Frontiers in Psychology*, 9, 2018.

[7] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10, 2018.

[8] Jack Cheng, Keyu Chen, and Weiwei Chen. Comparison of marker-based AR and markerless AR: A case study on indoor decoration system. 07 2017.

[9] Danilo Avola, Luigi Cinque, G.L. Foresti, Cristina Mercuri, and Daniele Pannone. A Practical Framework for the Development of

Augmented Reality Applications by using ArUco Markers. pages 645–654, 01 2016.

[10] Pablo Fernández Alcantarilla. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. 09 2013.

[11] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. Kaze features. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 214–227, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[12] Tony Lindeberg. *Scale Invariant Feature Transform*, volume 7. 05 2012.

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. volume 3951, pages 404–417, 07 2006.