IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 June 2024, accepted 25 July 2024, date of publication 6 August 2024, date of current version 16 August 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3439442*

SURVEY

# The Essentials: A Comprehensive Survey to Get Started in Augmented Reality

**FAYAZ MOHAMED HANEEFA[1,2], ABDULHADI SHOUFAN[1,2], (Member, IEEE), AND ERNESTO DAMIANI[1,3], (Senior Member, IEEE)**

[1]Center for Cyber-Physical Systems (C2PS), Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates
[2]Department of Computer and Information Engineering, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates
[3]Department of Computer Science, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates

Corresponding author: Fayaz Mohamed Haneefa (fayaz.haneefa@ku.ac.ae)

This work was supported by the Center for Cyber-Physical Systems at Khalifa University of Science and Technology.

**ABSTRACT** Augmented Reality (AR) has experienced a significant resurgence in popularity and interest in recent years. Despite numerous surveys and reviews in the field, information remains scattered and challenging to consolidate for newcomers. This paper addresses this gap with a comprehensive study of AR's state of the art. We begin with an introduction to AR and related terminologies. We then describe three enabling of AR and four enhancing technologies. A survey then covers different types of commercial AR hardware with detailed specifications and elaborations. We then address AR software and discuss its basic and advanced features, mapping them to software with feature matrices. Based on a thematic literature analysis, we extract a comprehensive set of guidelines for developing AR solutions into seven themes, from ideation to best practices for implementation and deployment. Finally, we explore some essential challenges in the field of AR. This paper serves as an essential resource for those looking to understand what AR is, what is needed to get started, how to approach development, and what the future holds for AR. By consolidating essential information and providing a solid foundation, this paper aims to help researchers and developers capitalize on the current cycle of interest in AR.

**INDEX TERMS** Augmented reality (AR), survey, tutorial.

## I. TABLE OF ABBREVIATIONS

- AR: Augmented Reality.
- AV: Augmented Virtuality.
- CGI: Computer Generated Imagery.
- DCC: Digital Content Creation.
- DOF: Degree Of Freedom.
- DOM: Document Object Model.
- ECS: Entity-Component System.
- EIS: Electronic Image Stabilization.
- EPnP: Efficient Perspective-n-Point.
- FOV: Field Of View.
- GPS: Global Positioning System.
- HMD: Head Mounted Display.
- HUD: Heads-Up Display.
- HVS: Human Visual System.
- IMU: Inertial Measurement Unit.
- IPD: Inter-Pupillary Distance.
- LFC: Lightform Compute.
- LiDAR: Light Detection And Ranging.
- MAR: Mobile AR.
- MR: Mixed Reality.
- MRTK: Mixed Reality Toolkit.
- NFT: Natural Feature Tracking.
- NLP: Natural Language Processing.
- OSC: Open Sound Control.
- OST: Optical See Through.
- PBR: Physically Based Rendering.
- PnP: Perspective-n-Point.
- POI: Points of Interest.
- PPD: Pixels Per Degree.
- PWA: Progressive Web App.

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

- RF: Radio Frequency.
- SaaS: Software-as-a-Service.
- SAR: Spatial AR.
- SDK: Software Development Kit.
- SLAM: Simultaneous Localization and Mapping.
- SMART: Seamless AR Tracking.
- ToF: Time-of-Flight.
- UE: Unreal Engine.
- URP: Universal Render Pipeline.
- UX: User Experience.
- VPS: Visual Positioning System.
- VR: Virtual Reality.
- VSLAM: Visual SLAM.
- VST: Video See Through.
- W3C: World Wide Web Consortium.
- XR: eXtended Reality, AR/VR.

## II. MOTIVATION AND CONTRIBUTION

This paper aims to provide a comprehensive perspective on the state of the art in Augmented Reality (AR). The aim is to help beginners understand the landscape of this technology, including hardware, software, design guidelines, and challenges. Advanced users and developers can also benefit from the comparative description of various enabling and facilitating technologies, including hardware and software, as well as from the design guidelines.

Hundreds of surveys and reviews on AR are available in the literature. However, these reviews primarily dedicate themselves to the use of AR in specific fields, including education [1], [2], [3], medicine [4], [5], engineering [6], [7], [8], [9], agriculture [10], tourism [11], and museums [12]. Most of these reviews aim to address AR validity, efficacy, and effectiveness [13], [14], [15], usability, user experience and emotions [16], [17], [18], [19], and opportunities and challenges [20], [21]. A few review papers touched on the hardware and software systems used by researchers on augmented reality [22], [23].

Thus, despite recent literature, understanding AR, the underlying technologies, the state of AR, and available commercial solutions is difficult and time-consuming. At this point, the holistic approach of two particular surveys on AR inspired us. Reference [24], published in 1997, still provides an excellent introduction to AR despite its age. It opened the narrative on AR, establishing the main concepts and their relations. Fourteen years later, another paper [25] provided a complete survey of the state-of-the-art of AR technology, systems, and applications. It described a more mature field populated by many industrial and academic research groups and the challenges of its time. The primary goal of this paper is to set a third act to these two narrations ( [24] and [25]), describing how some of the critical questions have been answered and providing an extensive and up-to-date starting point for researchers and practitioners. This paper is divided into the following sections:

1) Introduction to AR
2) Enabling Technologies
3) Enhancing Technologies
4) AR Hardware
5) AR Software
6) Guidelines for AR Development
7) Current Challenges in AR

The major contributions of this paper are as follows:

1) We have classified technologies associated with AR into *enabling* and *enhancing* technologies. Enabling technologies have further been classified into three building blocks of AR.
2) At the time of writing, we believe our paper has the most extensive collection of AR Hardware and Software ever listed in any AR survey. We have meticulously verified and listed the source of every feature and capability we could find.
3) We have included a list of features found in AR software with descriptions. Our descriptions will allow the reader to understand the capabilities of various software at a glance, using our *feature matrices* (Tables 3 and 4). The text maps the features of AR software to our descriptions for easier identification.
4) To our knowledge, this would also be the first paper to mention guidelines for AR development. This section is effectively the result of an informal thematic analysis (see Section XIV: Guidelines for AR Development). We have collated guidelines into groups according to the typical order of the development workflow.
5) We have also developed a ''litmus test'' to gauge AR applications worth pursuing.

We believe all this, coupled with sections on the fundamentals of AR, underlying technologies, and challenges of AR, should provide the best start to the field of AR.

## III. INTRODUCTION TO AR

This section aims to introduce AR through definitions and associated terminologies.

### A. THE DEFINITION OF AR

The most widely cited definition of AR comes from Ronald T. Azuma's seminal survey [24]. AR can be seen as *a variation of Virtual Reality (VR). In VR, the user is completely immersed in a virtual world. However, with AR, virtual elements are superimposed upon or composited with the real world.* In other words, AR *"augments"* reality, overlaying digital content onto the real world, while VR *"replaces"* reality.

Therefore, the first characteristic of AR is the ability to combine the real and virtual. This is why AR falls under the category of *Mixed Reality* (see Section III-B: Reality-Virtuality Continuum). However, combining the real and virtual is not limited to AR. Many feature-length films these days combine ''real'' live-action footage with ''virtual'' Computer Generated Imagery (CGI).

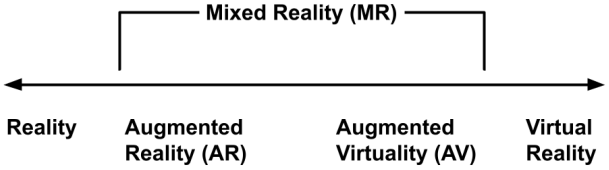


**FIGURE 1.** Various combinations of real and virtual content can be classified by the (simplified) reality-virtuality continuum defined in [28].

This leads to the second characteristic of AR: AR is interactive in real-time. Movies and videos that utilize CGI with live-action do not possess this characteristic. Therefore, one cannot classify them as AR. But what about live videos with virtual 2D overlays? Examples of such overlays would be those found on news broadcasts and are interactive in real-time.

This leads to the third and final characteristic of AR: AR is registered in 3D. The virtual objects must be registered in the 3D space of the real world. These three characteristics differentiate AR from other systems that combine the real and virtual.

In summary, AR can be defined as a system with the following three characteristics [24]:

- Combines virtual elements with reality
- Interactive in real time
- Registered in 3D

These characteristics define AR without constraining it to specific technologies, such as Head Mounted Displays (HMDs) [24]. This is important because the idea of what is real and virtual depends on the context (see Section III-B: Reality-Virtuality Continuum). Furthermore, AR is not necessarily restricted to augmenting visuals. It may very well be possible to expand AR to all senses [24]. Attempts have already been made to augment touch and hearing through haptic interfaces [26] and spatial audio [27], respectively. Unless explicitly mentioned otherwise, this paper's discussion will primarily revolve around visual augmentation.

### B. REALITY-VIRTUALITY CONTINUUM

In the previous section, we described AR as a system that combines virtual (digital) elements with reality. The converse is also possible, i.e., combining real elements into a virtual reality. This is referred to as "Augmented Virtuality" (AV). Taking VR as one extreme and reality as another, it is possible to place AR and AV on a spectrum, as shown in Fig. 1. This spectrum is often called the Virtuality Continuum. Since AR and AV combine the real and virtual, they fall under the broader category of "Mixed Reality" (MR) [28].

This classification, while simple enough, is not without issues. As seen with the characteristics of AR, classification based on how the real and virtual are mixed is not enough. Furthermore, AR and AV are differentiated by whether they are predominantly real or virtual. But what if this is impossible to distinguish, i.e., the experience is equally real and virtual? The terms AR or AV cannot be used in this case, but it can still be classified under the umbrella term MR.

There is still the problem of clearly defining the terms "real" and "virtual." In VR, anything generated by the computer is considered virtual. This gets more complicated in MR. For example, if we saw our arms with our own eyes, we would say our arms are real. But if we were to see our arm through a screen displaying our arm in real-time, we would also consider that to be "real". However, in the second example, our arm is nothing but a digital object generated by a computer. This creates an inconsistency concerning the definition of "virtual" that we gave earlier for VR. To prevent this inconsistency, we adopt the following revised definition [28]:

- Real: Anything with an actual objective existence
- Virtual: Anything that exists in essence or effect but without actuality

This distinction relies on the key terms actual, objective, and actuality, thus requiring these terms to be carefully defined. When the adjective "actual" refers to an entity, it signifies the state of its existence in reality. In a broader sense, however, "actual" can also relate to the true (genuine) nature of something as opposed to what is possible, hypothetical, or appears to be. Here, we consider the word "actual" in the latter sense. In turn, the word "actuality" refers to the fact of existing in reality. Finally, "objective" refers to the quality of being based on fact, not opinion.

Thus, "real" is any object that truly exists in reality, while "virtual" is anything that does not. A practical way to distinguish between them would be to ask if the object in question would exist without the AR system. Since they do not exist, virtual objects must always be generated by the system and cannot exist outside of it [28].

A more formal taxonomy for Mixed Reality displays is defined in [28]. The taxonomy is based on three dimensions, namely [28]:

- Extent of World Knowledge: It refers to the amount of known information about the real-world environment being displayed. This dimension ranges from completely unknown to completely known.
- Reproduction Fidelity: It refers to the realism of the display. This dimension ranges from completely unrealistic to completely realistic. It applies to both real and virtual elements.
- Extent of Presence Metaphor: It refers to the degree to which the viewer feels as if they are present within the displayed environment. This dimension ranges from completely outside looking in to completely inside the environment.

### C. OTHER DEFINITIONS AND TAXONOMIES

Another definition of interest is *eXtended Reality* (XR). XR is an umbrella term for AR and VR together. It also refers to technologies that support AR and VR capabilities. For this reason, the "X" in XR is also considered as the conventional

name for a variable instead of Extended and can be replaced with A or V for the context. Today, collectively experienced VR is often called a *Metaverse* where users interact within an entirely virtual environment.

There are more definitions and taxonomies regarding AR and other immersive technologies. They will be discussed as the need arises. This is to reduce confusion that arises from the number of definitions (see Section XV: Current Challenges in AR). For our current purposes, the characteristics of AR by [24], the Virtuality Continuum by [28], and the term XR are sufficient to understand AR.

## IV. ENABLING TECHNOLOGIES

This section will discuss the enabling technologies of AR and provide the necessary insight into its inner workings. Enabling technologies can be classified into three groups as follows:

- Registration
- Display
- Input

For easier reference, Fig. 2 provides a visual breakdown of the discussion structure in this section.

### A. REGISTRATION

For AR to be effective, the real and virtual objects in the world must be correctly aligned with each other. Otherwise, it breaks the illusion that the real and virtual coexist simultaneously. This problem of the need to align real and virtual objects with each other is called registration. Even as far back as 1997, registration was identified as one of the primary problems to be solved for AR to succeed [24]. Accurate registration is also necessary for specific applications, such as those used in medical procedures [24].

To solve the registration problem, keeping track of the position and orientation in the real world becomes necessary. Tracking systems can be implemented with Inertial Measurement Units (IMUs), Global Positioning Systems (GPS), Radio Frequency (RF) Signals, or even Acoustics. However, such systems can be vulnerable to signal degradation and interference. Active tracking systems also require sensors to be placed in the environment and be calibrated for use [29]. While this is possible, it may not always be practical. One reason is that this will constrain the user to a specific environment [29]. However, there is a more important reason for not relying on these methods for AR. Using information solely from the tracking system leads to the issue of having an ''open-loop'' controller [24]. While the system may be able to provide information about the real world, it cannot know whether the real and the virtual match. There is no feedback for correcting any issues with alignment. In addition, the sensors themselves can be prone to errors. IMUs, for instance, suffer from the problem of drift. The reported values drift over time due to accumulated errors.

It is, therefore, better to have a ''closed-loop'' controller where there is some form of feedback in the system [24]. This led to the use of video-based approaches to aid in registration. It becomes possible to detect features in the environment using image processing or computer vision techniques, which are used to enforce registration. In many cases, multiple cameras are used to increase the available information and gain a better perspective of the environment. A few visual-based techniques of interest are discussed below. This will provide adequate insight into some of the techniques utilized to allow AR systems to understand their environment. The following techniques need not be used in isolation; they can be combined and used with each other. Since the systems for registration ultimately involve tracking the environment or something in it, these techniques can also be utilized to provide certain features in modern AR software. These features can be seen in this paper's sections for Basic and Advanced Features.

Before we begin, registration in vision-based AR can be considered as a pose-estimation problem [29], [30]. This can be better understood through how the real and virtual objects are combined in AR. A transform of an object contains its translation (position), rotation, and scale. On the real side, the world can be given an origin point of our choice. The camera capturing the real has a transform relative to the world origin. Similarly, there exists a virtual world with its origin and a virtual camera with its transform. AR is achieved by matching the transforms of both worlds and cameras. This allows the system to display the virtual objects correctly over the real world by matching the transform of the real and virtual cameras [30]. Since both cameras operate on the same scale, the scale can be ignored. Thus, the challenge is estimating just the position and rotation/orientation, called the pose [29].

#### 1) PERSPECTIVE-N-POINT (PNP)

As discussed above, we are dealing with a pose estimation problem, trying to match orientation and position in 3 dimensions. However, cameras typically only return 2D images, which adds to the problem. PnP is one method that can be used in this situation. PnP works by taking an image of an object/scene and identifying the coordinates of some key points on this object. It then compares the coordinates of these points from the image to a 3D model of the object. By finding the correspondence between the key points in the 2D image and 3D model, the object's pose can be estimated [30].

It should be noted that PnP is the name of the problem, not the solution. There are different approaches to solving a PnP problem. One is P3P, where three key points are utilized for pose estimation. Another is Efficient PnP (EPnP), which is used for solving problems where the number of key points (n) is greater than or equal to 4.

PnP requires 3D models of real-world objects to function. Thus, when considering PnP, one must consider how to obtain these models. It also requires key points on the objects, without which the correspondence between 2D and 3D

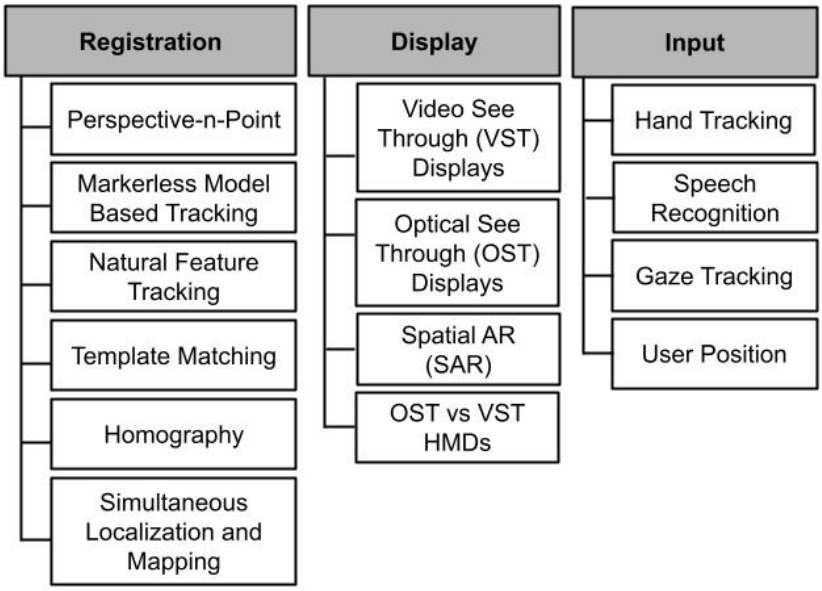


**FIGURE 2.** A visual breakdown of the section "Enabling Technologies". Primary topics are at the top (gray) and go from left to right.

cannot be established. In other words, PnP requires "natural or intentionally placed features (fiducials) whose positions are known a priori" [29].

#### 2) MARKERLESS MODEL-BASED TRACKING

The premise of Markerless model-based tracking is to track the pose of an object without the need for markers in the scene or key points on the object. This method works by extracting features such as the edges and contours of the object from the image. It then attempts to match the identified features with that of the object. The pose is estimated by aligning the features [30]. As with PnP, this technique also requires a 3D model of the object to function.

#### 3) NATURAL FEATURE TRACKING (NFT)

For a visual registration system, dependence on known feature points can pose the following challenges [29]:

- Limit operation to regions where at least three known features can be seen unobstructed
- Reduced pose estimation stability with fewer visible features
- The known features do not always correspond with the desired points

NFT is one method that can overcome the challenges mentioned above. It identifies natural features (points and regions) in the environment, which are used to estimate the camera's pose. Since it can automatically identify features within the environment, it does not suffer from the challenges above. As it does not need to rely on known points, the system can estimate pose beyond known areas or in scenarios where known feature points go out of view [29].

#### 4) TEMPLATE MATCHING

Template Matching attempts to embed the pose estimation into image processing itself. The system uses a reference image, called a template, that effectively acts as a 2D model. The pose is estimated by comparing the current image with the reference image and accounting for the motion and warping for the image seen in the camera feed [30].

#### 5) HOMOGRAPHY

Any two images of the same planar surface in space are said to be related by what is called homography. Similar to PnP, this method works by utilizing correspondences between points. However, the correspondence is between two images of the points on the same planar surface. The correspondence can be used to calculate what is called a homography matrix. This homography can then be used to calculate the pose [30].

This is a simple and efficient method for determining the pose of a camera in relation to a planar surface, as it only needs to solve for the camera's translation and rotation with respect to the plane. However, it has some limitations. It is sensitive to noise and outliers and can only work for planar scenes.

#### 6) SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)

So far, we have seen two techniques that require 3D models to function. However, 3D models are not always available. Practicality is, therefore, limited by constraining the system to specific scenarios and environments where 3D models are available. While some methods do not require them, there is a case to be made for using 3D models. The registration problem ultimately arises from the need to maintain the immersion between real and virtual. By having a 3D model of

the real-world environment, we can make the virtual objects interact with the real world in realistic ways.

For example, let us say we were to throw a real ball into a room with furniture. It would bounce off the walls, destroy a lamp, go out of view behind some furniture, come rolling out into view, and finally stop. A 3D model of the environment makes it possible to simulate the same physical interaction for a virtual ball thrown into the environment. In 3D, systems replicating real-world physical interactions are called physics engines/libraries. From here on, the word physics in the context of a virtual object refers to simulated physical interaction and related physical properties of the virtual object. Since the system has a 3D environment model, it can calculate physics even in spaces that cannot be seen, like the region behind furniture. Outside of the virtual ball being incapable of breaking a lamp in the real world, this system will still provide a great sense of realism and immersion.

So, we would like to have 3D models without having the availability of premade models constraining the system. This is where we can utilize SLAM [30]. Like PnP, SLAM is the name of the computational problem. SLAM originated in robotics. The problem lies in mapping an unknown environment while simultaneously trying to track the location of the entity moving through it. This leads to a conundrum. One would need a map of the environment to identify one's position. However, one needs to keep track of one's position to map the environment.

Nevertheless, SLAM does have solutions and they have been implemented in many practical applications. Commercial systems often refer to this by the name of spatial mapping. There are many variations of SLAM, utilizing different sensors, sensor configurations, and SLAM algorithms while providing maps of various formats. Discussing them all would require a survey of its own. However, when considering sensors, two methods of note are Visual SLAM (VSLAM) and Light Detection And Ranging SLAM (LiDAR SLAM). Visual SLAM utilizes images from cameras and other image sensors. LiDAR SLAM utilizes a LiDAR system that uses lasers to measure distances with high precision. These variants are important as the sensors they need are readily available in modern hardware (see Section VI: AR Hardware).

### B. DISPLAY

With the problem of registration solved, the next step would be to figure out how the virtual objects can be displayed. But before discussing displays, it would be helpful to understand a few important properties of the Human Visual System (HVS) relevant to AR.

Let us first refresh how the HVS functions. The first requirement for visual perception is light in the correct wavelength range (from 390 to 750 nm). Light enters the human eye through the pupil and is focused on the retina at the back of the eye by the lens. The retina has light-sensitive receptors, namely rods and cones. These receptors respond to light and send signals to the brain, resulting in vision. The rods are more sensitive to light, contribute little to color perception, and react slowly to changes in light conditions. The cones are responsible for color vision and respond faster to help us perceive finer details [31].

The first important property of the HVS is that the retina does not have a uniform distribution of rods and cones. The retina has a small area called the macula that is used to perceive a sharp image. There is a smaller area called the fovea that has the sharpest vision. Therefore, the resolution of the human eye is not uniform. Second, since the cones are not distributed equally, color sensitivity is also not uniform across the human eye [31].

Third, the HVS has a visual field or Field of View (FOV), a limited area in which it can perceive. The vertical FOV extends 50° to 60° up and down the horizon, while the overall horizontal FOV can go up to 180°. The FOV is known to decrease with age. Fourth, the HVS can perceive latency and flicker but only up to a limit. The perception of latency and flicker needs to be reduced for a smooth experience and can be done so by coming closer to the limits [31].

Finally, the HVS supports depth perception, allowing us to perceive the position and sizes of objects. The HVS has accommodation mechanisms to keep objects at various distances in focus. This is further supported by vergence. Vergence is the movement by both eyes that allows for binocular vision. Vergence supports depth perception by triangulating the positions of objects in each eye with the Inter-Pupillary Distance (IPD), which is a fixed distance between the eyes. The IPD is variable from person to person. In the physical world, objects are hidden/occluded by other opaque objects placed in front. This is called occlusion. Occlusions contribute significantly to depth perception as a visual cue by showing where objects lie relative to each other [31]. The HVS has more properties of interest, but the five listed points are more than adequate to understand what AR systems should address for an excellent visual experience.

There are many approaches to AR displays, but they can be categorized into three groups [31]:

- Video See Through (VST) displays
- Optical See Through (OST) displays
- Spatial AR (SAR)/Projection Mapping

#### 1) VIDEO SEE THROUGH (VST) DISPLAYS

VST displays, also called Pass-Through Displays, use a camera to capture the real world through video. This video is overlaid with virtual objects and then displayed (or passed through) to a screen. The real world is seen indirectly by the user through the screen. As discussed before, the advantages of video-based registration led to its adoption in practical use. As such, the cameras needed for VST displays are already present in many systems.

A ubiquitous and accessible example of a VST display would be a smartphone or tablet. However, there are a few practical issues. It does not leave the hands of the users free

for other tasks. The field of view is also minimal, constrained to their relatively small displays. HMDs address these issues. As the display is worn on the head, the display will always follow the head's orientation, leading to a more immersive experience. In addition, AR headsets can free up a user's hands, making AR more practical for specific applications. However, the need for the headsets to be light, ergonomic, and mobile adds many constraints.

#### 2) OPTICAL SEE THROUGH (OST) DISPLAYS

OST displays consist of a transparent display through which the user can see the real world. The virtual images are then superimposed into the user's view. The exact means by which the combination is done may vary. Still, it typically involves a light source (to project the virtual image) and a system to adjust the projected image (such as lenses or mirrors). The optical combiner is the system that combines real-world light with the projected virtual light. The displays may be made reflective to improve the perception of virtual objects. Such displays would appear semi-transparent as if looking at the world through sunglasses. Examples of OST displays are Heads Up Displays (HUDs) and Head Mounted Displays (OST-HMDs).

HUDs originated outside of AR; back then, they could be seen as AR without registration. It was simply a means to overlay additional information over the real world. It is commonly used in aircraft, allowing pilots to view crucial information without having to take their eyes away from the view of the world outside. They usually consist of a flat piece of glass that acts as the display.

OST-HMDs retain the mobility and freedom that can be found in VST-HMDs. However, unlike VST displays, the user can see the real world in full detail with OST. It is not dependent on the resolution of a pass-through camera. In this regard, OST-HMDs are more akin to spectacles. However, the visual fidelity of virtual objects is usually the trade-off that needs to be made with these systems.

#### 3) SPATIAL AR (SAR)

The final method of display we consider is SAR. Projectors are utilized to display the virtual objects in the real environment directly. The user is not required to hold or wear a display system. The system is usually static and confined to an environment. Since the graphics are directly projected onto the real world, the visual fidelity of the virtual objects is affected by the inability to control the appearance of the real world [31], [32]. Additionally, as the user may come between the projector and the display surface, there are challenges to interactivity. SAR has one possible advantage over other solutions. Since the experience is directly projected onto the environment, it is independent of the user. This allows for a shared session without the need to sync between multiple devices. It also removes the need for every user to have an individual device to join the experience.

The display technique used here is called Projection Mapping [31]. The term is often interchangeably used with Spatial AR. However, plenty of applications utilize projection mapping without being interactive. As such, it is best to reserve the term projection mapping for the display technique rather than an AR system.

#### 4) OST VS VST HMDs

One advantage of OST over VST is the resolution of the real world. With VST, the resolution of both real and virtual is limited to the resolution of the camera and the display [24]. The resolution of most displays is far less than the natural resolution of the human eye. While OST displays cannot match the resolution of human eyes for virtual objects, it does not degrade the view of the real world [24], [31]. Furthermore, with VST, the cameras act as the eyes of the user. However, these cameras are not located at the exact position of the eyes. Even the distance separating the cameras may not match the IPD. The result is that the camera's view does not match the user's normal vision. Thus, the images must be adjusted with VST [24].

The transparent nature of OST displays can provide minor advantages in certain scenarios. Maintaining eye contact and reading other people's facial expressions is possible since the display is transparent. This may be of importance in social settings [31]. Additionally, if the power cuts out, the user can still see in the case of an OST-HMD [24]. However, this is not too much of an issue, as today's more ergonomic HMDs can be easily removed. Such a feature may become more important in scenarios where visibility is critical to safety.

VST has multiple advantages over OST because it displays a digital stream of the real world. One basic obstacle with OST is that the virtual objects cannot occlude real objects. As discussed before, occlusion is a very important perceptual cue. It allows users to understand the general depth of a scene and, thus, the positioning of objects (both real and virtual) in 3D space. The optical combiners in OST allow light from both real and virtual sources. For proper occlusion by virtual objects, blocking out the light from a real source is necessary. Without it, the virtual objects will perpetually appear semi-transparent [24], [31]. While solutions exist, we could only find one commercial OST solution at the time of writing (see Section VI-B2: Magic Leap 2, under AR Hardware: HMDs). Their complexity makes them difficult to implement. With VST, both real and virtual images are available in digital form. Both images can be composited to include the real, the virtual, or something in between. The transparency of the virtual objects can easily be controlled, allowing for occlusion. This also enables VST displays to match better the brightness, color, and contrast between the real and virtual [24], [31].

OST displays can provide an instantaneous view of the real world, unlike VST displays. While this may seem like an advantage for OST, it is advantageous for VST. The system requires some time for registration and rendering.

As such, the virtual image stream can be slightly delayed compared to the instantaneous real-world view. This creates a temporal mismatch, as the real and virtual objects do not align with time. Since VST controls both the real and virtual image streams, it can delay the real stream to sync with the virtual stream [24]. Thus, VST displays can, in theory, achieve visual coherence to a better extent than OST displays. This is despite the real-world resolution issue of VST displays.

Furthermore, OST displays rely on optical principles. Thus, they must correct any distortions with the image optically rather than digitally. HMDs need to be simple, cheap, and lightweight for practical use. These constraints limit what can be done optically with OST displays and leave them with a reduced field of view. On the other hand, VST displays can correct any image distortions digitally [24]. It may be computationally intensive, but this is less of an issue with today's computational advances.

### C. INPUT

The user requires some form of input to interact with an AR system. Theoretically, any form of input that the system can recognize can be utilized. This could be the typical input method used by a platform. For example, touchscreen displays are available for hand-held devices, and a keyboard and trackpad combo is available for laptops. Hand-held controllers like those used in VR could also be utilized in the case of AR headsets. However, such input systems do not leave the hands of the users free. It can also reduce the level of immersion in some scenarios, particularly with AR headsets. Therefore, these headsets often come with different inputs that are more suited to meet these demands.

#### 1) HAND TRACKING

Hand tracking is utilized in AR to provide a more natural way to interact with the system. Many interactions in real life include the use of hands. They are used to interact with the physical world by manipulating real-world objects. They can also be a means of communication through gestures. Our hands provide a natural and significant way of interacting with the world around us [33]. Since AR aims to combine the real and virtual, utilizing the same means of interaction for both real and virtual objects is intuitive.

It is possible to simplify the challenge of hand tracking by utilizing certain techniques from the field of motion capture. One method would be to use sensors that can be worn on the hand. IMUs, for instance, could be utilized to calculate the orientation of the fingers and the palms. Yet another method would be to place markers on a hand that cameras can track. However, these methods can burden the user and lead to unnatural movements. They also require calibration and setup to obtain precise measurements [33]. These methods are suited for scenarios where the environment can be controlled, and the calibration and setup are worth the time spent. An example would be a motion capture studio. However, AR systems, particularly commercial headsets, must be used in various environments. Time spent on calibration and setup can affect the usability and practicality of the AR system. These same issues apply to registration when utilizing such tracking systems.

Considering this, markerless vision-based techniques would be the best option for AR. It is possible to track the motion of the hand through image alone (similar to registration). However, there are significant challenges to this approach. Visual tracking techniques, in general, are affected by the quality of the image. A lack of contrast between the hand and the environment can make it difficult to distinguish the hand. The hand is also particularly prone to occlusion issues as the fingers can be hidden behind other fingers or the palm. A human hand is also quite complex to track due to amount of joints and the possible rotations for those joints. In addition, interactions can involve manipulating a real object. This can complicate the tracking further through occlusions [33].

The power, size, and weight constraints of HMDs add to this problem. There is a limit to the number of cameras and where they can be positioned on the HMD. In many commercial AR headsets, the system can only track hands in a limited area in front of the user. Considering these challenges, most systems implement a gesture system. The user's hands are compared to those in a gesture library. Depending on the gesture, a certain function will be activated. For example, users could pinch their hands to grab a virtual object. Once the system recognizes the gesture, it identifies the virtual object the user has ''grabbed'' and ties its position to the user's hand. The system tracks the user's hand as it moves and updates the position of the virtual object to match. Thus, the system creates the illusion of the user grabbing and moving around a virtual object. Using this system reduces the complexity of the tracking problem. Interestingly, while this system limits the natural interactions of the user, it can also make the system easier to use. As the library of gestures is limited, the user has less to learn regarding interactions. More importantly, using the same gesture library ensures that users can use the same gestures in different AR applications and expect the same behavior.

#### 2) SPEECH RECOGNITION

There will be certain scenarios where the user's hands cannot be used for interaction. Perhaps the AR application assists the user with a physical tool that requires both hands. The user thus requires a different mode of input where they can keep working without taking their hands off the task. Speech recognition is one solution for such scenarios. The system can act as a digital assistant, performing the actions on the user's behalf upon request.

#### 3) GAZE TRACKING

Speech recognition is not always a viable option. Perhaps the environmental sounds are too loud, or there is a requirement to remain silent. This is a bigger problem if hand tracking is

also not possible. While the potential means of interaction have been greatly reduced, head and eye tracking can still provide some input. The idea here is to utilize the user's gaze as input. The orientation information from registration can be combined with the orientation of the head or the eyes. Thus, the system can determine the virtual object under the user's gaze. Hence, we can collectively refer to such a form of input as gaze tracking.

It is possible to simulate gaze tracking on handheld devices. By assuming the device to be the ''head'' and the rear cameras as the ''eyes,'' we can take the device's orientation as the direction in which it gazes. In theory, further refining the tracking may be possible by implementing eye tracking with the front cameras. Since the user must observe the screen in such devices, the front cameras could be used to track which object the user views on the screen.

A limitation of this mode of input is that the only variable the user can control here is the orientation of the gaze. If gaze tracking is the only available input, it creates a problem. Since the gaze is used to identify a target object, there is no form of input left to interact with said object. This can be counteracted by making any remaining interactions automatic and context or object-specific. For example, an object can be selected by maintaining the user's gaze on it for a certain amount of time, with the timer managed by the application. As another example, the system can automatically scroll text on a virtual screen depending on whether the user's gaze is on the upper or lower bound of the screen [34]. Gaze is best used with voice commands or other input modalities where gaze provides the user's intent [34]. Finally, with greater precision than head tracking, eye tracking can be used for text entry with a visual keyboard, especially if other input modalities are unavailable [34].

#### 4) USER POSITION

The position of the user can be used as a form of input. With an HMD, the user's position corresponds with the device's. With a handheld, the system has to assume the position of the handheld as that of the user. Like gaze, user position is limited in a standalone scenario, but it could be useful as supplemental input for other modalities. The position can be processed into additional information, like motion or proximity. AR content can then react according to the user's motion or proximity to the content. For instance, a virtual flower could only bloom if the user approaches it. Or a virtual character's gaze could be made to follow the user as they move around.

## V. ENHANCING TECHNOLOGIES

This section discusses other technologies of interest. They are horizontal technologies as they have applications in various fields and, thus, are not developed with AR in mind. Nevertheless, they can significantly enhance the capabilities of any AR system, and attention must be paid to their advancements and challenges.

### A. ARTIFICIAL INTELLIGENCE

There is a need in AR for systems that can better comprehend their surroundings and the user's actions. This need can be seen in the discussion of enabling technologies. If an AR system could understand the environment it sees, it can lead to better tracking and registration. If the system could understand the motion of the user's hands or their speech, it could lead to a more natural means of input. AI can help in this regard massively.

Various methods and techniques in AI have already found their way into AR. The techniques previously discussed for vision-based registration fall under Computer Vision, a sub-field in AI. SLAM, in particular, has found applications outside of AR in autonomous machines that navigate environments. AI can improve registration through scene semantics (see Section IX: Advanced Features). AI can also be applied in AR through Speech Recognition and Natural Language Processing (NLP). This can add the capabilities of a virtual assistant to any AR system. Finally, AI can also be used in motion tracking to predict the motion of a tracked body when it goes out of view. This could have potential applications in improving hand tracking. As discussed before, the hand is particularly prone to occlusion issues. AI may not be able to solve this issue entirely, but it can provide improvements. Since AI is a horizontal technology, its inclusion with AR need not be limited to improving just registration and input.

Some challenges need to be addressed when using AI. Training an AI can be significant, constrained by available data and computing resources. They can be computationally intensive to run. AI models can also be black boxes with unknown reasoning for the provided output. Social challenges also apply. AI has applications in multiple fields, but legal regulation has not caught up with the advancements. The massive amount of data that has enabled the rise of AI also leads to arguments on user privacy.

### B. REMOTE COMPUTING

The massive growth of smartphones, tablets, and other mobile devices has made AR accessible to the masses. They possess computing power, cameras, and various sensors that make AR possible. In conjunction with headsets, these devices constitute what is classified as Mobile AR (MAR). The three characteristics of AR define MAR, but with an additional condition that the AR application runs or is displayed on a mobile device [35], [36]. MAR has seen a lot of attention from both industry and academia. This is partly due to the accessibility (affordability) of such devices and advances in their computing power. Another reason is that mobile devices allow AR applications to follow the user and be more immersive. But this source of strength is also its weakness. The need to be mobile adds constraints to the weight and volume of a mobile device. A device that is too heavy or bulky to be carried around is too impractical to be a mobile device. Trade-offs need to be made with mobile devices. Consider

the battery, for instance. Constraints on size and weight limit the size of the battery. This limits the total available power, limiting the unit's computational power and operational time. There are other constraints to be considered as well. Increased computational power can also mean an increased thermal output. The small form factor adds challenges to possible cooling solutions. Additionally, since mobile devices are to be held or worn with AR, the devices should also be kept cool enough for continued human contact.

The constraints of mobile devices lead to a design with limited computing power. However, there are reasons why increased computing power is still desirable. An AR system can provide a more immersive experience if it has the increased computing power to support it. As discussed in the previous section, AI is a good example. Additionally, increased computing power allows the system to render more complex virtual objects with greater detail. Additional computing power also ensures that the necessary frames are generated on time, even after all the processing and rendering.

Improvements in hardware and software can help overcome the limitations of limited computing power. However, another solution to this problem, especially if such power is needed right now, is to utilize remote computing. The idea is to offload the heavier computational tasks to another computer, which will then return the results to the MAR device. It essentially removes the limitation on how much computing power a mobile device can have. This can also remove the need for powerful hardware on the device itself, reducing the cost and complexity of AR devices and making them more accessible to a wider range of users. Another advantage of remote computing for AR is the ability to access data and resources from anywhere with an internet connection. This allows users to access these resources remotely rather than storing them locally on the device. Finally, remote computing can help to improve the scalability and reliability of AR applications. By using cloud-based resources, AR applications can be designed to handle large numbers of users and fluctuations in demand without additional hardware or infrastructure. This can help to ensure that AR applications are always available and perform well, even under heavy load.

Remote computing need not be constrained to the Cloud or the Edge. Even local devices could be used to raise the available resources for a mobile system. It is even possible to combine them to obtain hybrid architectures [36]. Each setup can have its unique challenges. However, the common trade-off between them is a new set of challenges concerning the network between the mobile device and the remote computer (see Section V-C: Computer Networks).

### C. COMPUTER NETWORKS

The ability to communicate between two separate devices is a major factor in their viability of functioning together. There are two primary considerations for networks when incorporating remote computing with AR: bandwidth and latency. A higher bandwidth will allow the transfer of larger volumes of data. A remote computer could render extremely high-resolution images for a display, but it does not matter if it is too much to be sent to an AR device in time. AR is also a real-time system. There is already a delay in systems from input to output. Utilizing remote computing can add to this delay, which will appear as latency. It is important to maintain low latency to avoid issues such as temporal mismatch. The specific challenges with computer networks relating to bandwidth and latency will depend on the remote computing architecture used [36].

### D. HAPTICS

"Haptic devices enable human-computer interaction through touch and external forces" [26]. Such technology allows users to experience tactile and kinesthetic sensations. Tactile refers to sensations of touch or the skin. Kinesthetic or proprioceptive perception is the sense of awareness of the body's state. This includes the position, velocity, and force supplied by the muscles [26]. By expanding to another set of senses beyond the audio and visual, such technology can further add to the immersion and realism of AR. Users can touch and feel virtual objects by providing appropriate haptic feedback. Haptic feedback can also be used with or as an alternative to other forms of feedback. For example, it can be used with GUI interactions in addition to the visual feedback of the UI. It can also be used as an alternative with visually impaired people, who may struggle with visual means of feedback [26].

Haptic devices can be classified into two types according to the specific sensation they activate [26]:

- Tactile/Cutaneous
- Kinesthetic/Proprioceptive/Force

The devices can be further classified according to their underlying technology or form factor. For simplicity, this paper will stick to the umbrella term of haptics. While there are many commercial solutions for haptics, they are not necessarily for AR. Thus, integrating such solutions into an AR project may require additional effort, resources, and developer support. The most accessible solution would be through smartphones or similar handheld devices. Haptics have become a standard in such devices, and the devices have become ubiquitous. Controllers such as gamepads, racing wheels, or those of XR devices can also come with haptics built in.

The previous devices have haptics as a secondary feature to enhance their overall function/experience. However, there are commercial solutions built with the primary goal of delivering haptic feedback. One class of devices is meant for teleoperation or similar scenarios. They are desktop-based (thus immobile/static), with the device delivering haptic feedback for the user's interactions on a screen. Touch and Touch X by 3D systems [37] and Desktop 6D by Haption [38] are examples of such devices. They are typically used for manipulating and feeling on-screen 3D objects, which are useful for applications such as 3D modeling. Another

interesting example would be the Novint Falcon. It is a game controller focusing on haptic feedback for greater immersion. For example, the attachable pistol grip could allow players to experience recoil for a gun fired in a game [39]. Novint Technologies appears to be defunct [40], but the Falcon controller can still be found for sale [41].

One drawback with the devices covered so far is that they occupy the user's hand and do not leave them free for the user to manipulate the real world. This can be addressed with wearable or mid-air haptic devices. Many wearable solutions can be found in the field of VR. The Senso glove [42], TactGlove [43], and HaptX G1 Gloves [44] are some examples of haptic gloves. It should be noted that HaptX G1 utilizes air and requires an "Airpack" that can be placed or carried on a user's back [44]. The Senso Suit [42] and the TactSuit Series from bHaptics [43] are some examples of wearable devices for the torso. bHaptics also provide solutions for other sections of the body, such as the TactVisor for the face and Tactosy for feet, arms, and hands (excluding fingers) [43]. Similar to hand tracking, a particular challenge of wearable haptic devices is that they can restrict the user's ability to interact with the real world naturally. A bulky haptic glove, for instance, could make it hard to feel or grasp a physical object. Mid-air/contactless haptic devices can provide haptic feedback without contact. The user is freed from having to wear any device. However, mid-air haptic devices have a limited workspace, constraining the user to remain near the device. No commercial solutions could be found, but Ultraleap is working on a product using an array of ultrasonic speakers [45].

The industry has been moving towards implementing high-fidelity haptic feedback into devices where haptics have traditionally taken a backseat. This trend has been referred to as HD haptics and attempts to deliver more distinct and detailed haptic feedback on devices rather than a simple buzz or vibration [46]. Interhaptics is a tool developed to help aid the integration of such haptics into projects. It is a cross-platform solution, providing a common foundation to design, test, and deploy haptics across a range of devices. It is available as an SDK, with integrations for Unreal Engine and Unity [47].

## VI. AR HARDWARE

This section will cover some popular hardware solutions available for AR applications. Solutions available for commercial use or general consumers, ideally with publicly available software for development, were preferred for this section. The hardware in the market can be divided into three categories:

- Smartphones and Tablets
- Head Mounted Displays (HMDs)
- Spatial AR (SAR)

HMDs have been further divided into a subcategory of "AR Glasses". For easier reference, Fig. 3 provides a visual breakdown of the discussion structure in this section.

### A. SMARTPHONES AND TABLETS

The wide array of sensors, cameras, and improvements in computing power make these mobile devices even more suitable for AR. AR here is always implemented as VST displays. Unless transparent phones or other such devices become a reality, OST displays are not possible with this class of devices. As discussed previously, these devices limit the user's movement and have a limited field of view (see Section IV-B: Display under Enabling Technologies). The primary reason to consider this category would be their widespread adoption. Any AR application developed for these devices can be distributed to a large market of users.

Care should be taken, however, to ensure that applications can run on the various combinations of operating systems and hardware. Devices can have different capabilities when it comes to sensing or processing. To account for this, Software Development Kits (SDKs) provide means to identify the device, its OS, and its hardware capabilities. Any feature that the device cannot support can be disabled through software.

This category can be divided into two groups based on the OS: iOS and Android. While there are more operating systems, these two have the largest market share. More importantly, they both possess their own AR SDK, simplifying development.

#### 1) iOS DEVICES

One advantage that iOS enjoys over Android is the comparatively limited set of devices it needs to support. iOS is better attuned to its hardware as a result. LiDAR sensors have also been added to Apple's more recent offerings. This provides their devices with depth information, significantly improving registration. AR applications can be built for these devices with the ARKit SDK. At a minimum, ARKit requires iOS 11.0 and an iOS device with an A9 processor to function [48]. The exact list of supported features will vary by model. To learn more, see ARKit under Section: AR Software.

#### 2) ANDROID DEVICES

Unlike iOS, Android needs to support a far greater variety of systems. Supported AR capabilities can easily vary from device to device. Applications can be made using the ARCore SDK, which Google maintains and develops. ARCore maintains a list of supported devices [49]. To see the AR features of this SDK, see ARCore under Section: AR Software Frameworks.

### B. HEAD MOUNTED DISPLAYS (HMDs)

The "Sword of Damocles" by Ivan Sutherland is an HMD often cited as the first demonstration of AR [31]. This is debatable, as claims of similar systems predate Sutherland's device. "The ultimate display," as Sutherland termed it, is also often cited as the first VR headset. Nevertheless, HMDs have been a significant aspect of AR. It even led to AR being defined solely on HMDs, a problem addressed by Azuma's definition of AR [24]. The possibility to

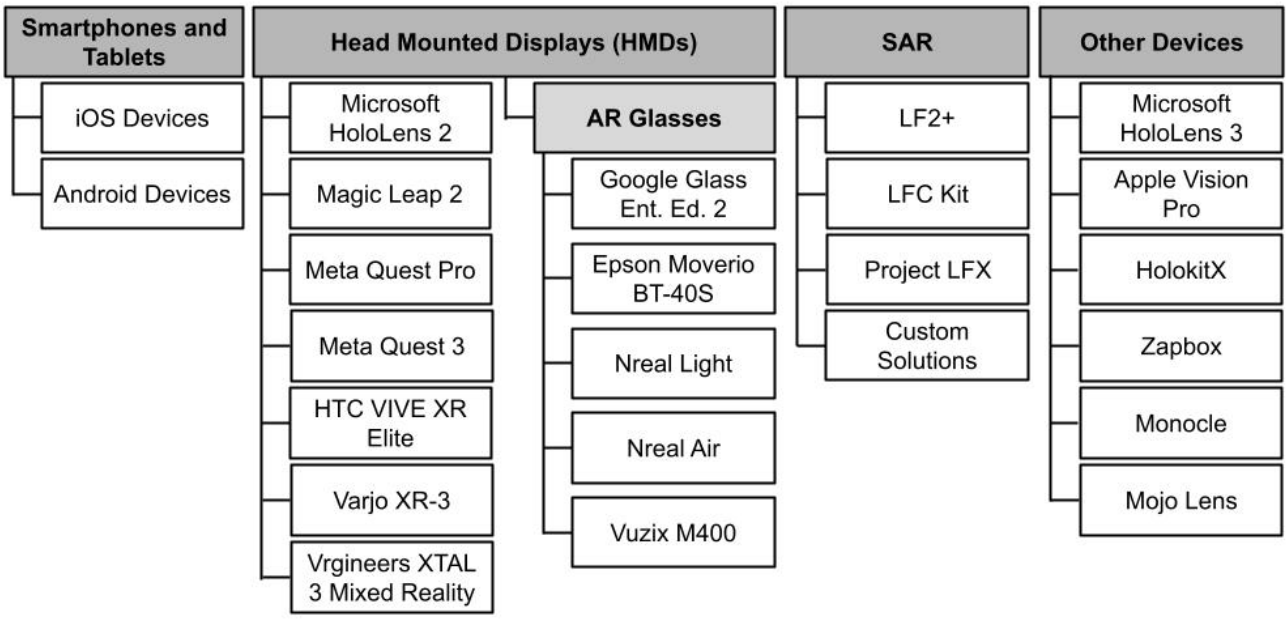


**FIGURE 3.** A visual breakdown of the section "AR Hardware". Primary topics are at the top (gray) and go from left to right.

look around, move around, and experience AR in various environments were the likely factors that led to the appeal of HMDs. Both OST and VST displays have been considered for HMD-based AR systems. For the pros and cons between them, see Displays: OST vs VST displays in Section: Enabling Technologies. Refer to Table 10 (see Appendix) to easily compare the specs between the HMDs listed in this section. Refer to the respective sections for more details on the specifications marked with an asterisk. For easier reference, the specifications from the table have been marked in bold in the text.

#### 1) MICROSOFT HOLOLENS 2

The Microsoft HoloLens 2 is a **standalone OST HMD**. The name HoloLens likely comes from the word "hologram", which Microsoft uses to refer to the virtual objects displayed by this HMD [50]. This is partly due to the OST display, resulting in semi-transparent objects. Thus, they appear like the holograms portrayed in fictional media, as see-through 3D objects displayed in 3D space. The HoloLens 2 has an **FOV** of **43° horizontal** and **29° vertical**, which gives **52° diagonally**. The **resolution per eye** is 1440 × 936 [51].

The refresh rate of the HoloLens 2 is complicated. It is missing in the official specifications [52]. The refresh rates from third-party sources vary greatly, reported as 60Hz, 75Hz, 90Hz, or 120Hz. Microsoft documentation mentions both 120Hz and 60Hz [53], [54]. The display resolution is specified as 120Hz under HoloLens 2 Display Troubleshooting [53]. However, 60Hz is listed as the refresh rate under "hologram" stability [54]. It is best to settle on **60Hz as the refresh rate**, as Microsoft recommends targeting 60 FPS for the stability of virtual objects. It helps minimize the overall latency, maintain consistent latency, and reduce judder to maintain a smooth user experience [54].

The HoloLens 2 has a suite of sensors to support registration and tracking. It has **4 visible light cameras** for **6 Degree of Freedom (DoF) head tracking** along with an **IMU**. This, combined with a **1-MP Time-of-Flight (ToF) sensor**, allows the headset to perform Spatial Mapping. The HMD can create a real-time rough 3D model (mesh) of the environment. This allows for more immersive AR applications (see Section IV-A6: SLAM under Enabling Technologies: Registration). The HMD also supports hand tracking with a two-handed, fully articulated model. **2 IR cameras** provide **real-time eye tracking**. They also provide security with iris recognition. Finally, the HoloLens 2 also has an **8-MP RGB camera**, capable of **1080p video at 30 FPS**. This is useful for capturing the real environment that the user can see. For **audio**, it has a **5-channel microphone array** and **built-in speakers for spatial audio**. Voice commands are possible with the standalone HMD, with NLP requiring internet connectivity [52].

The headset is powered by the **Snapdragon 850 mobile compute** platform with passive cooling. It has **4GB DRAM** and **64GB for storage**. For connectivity, it supports **Wi-Fi 5 (802.11ac** 2 × 2) and **Bluetooth 5.0**, with **a USB Type-C port** for physical connections and charging. Under active use, it has a **maximum battery life of 3 hours**. As for ergonomics, the HMD can **fit over the user's eyewear** and **weighs 556g** [52]. The HMD is single-size, but the headband can be adjusted to fit your comfort. The visor can also be tilted to accommodate eyewear better. An overhead strap can also be optionally used, providing more comfort over long periods of use [55].

The HoloLens 2 predates many of the HMDs listed in this paper and is often outclassed by them on hardware, particularly on FOV and compute. But thanks to its maturity, the HoloLens 2 has a rich software ecosystem to support it. The HMD runs on Windows Holographic OS and supports various applications and services from Microsoft like:

- Microsoft Edge: Web browser, also supports WebXR (see Section X-B: WebXR under AR Standards)
- Dynamics 365 Remote Assist: Real-time remote collaboration and assistance
- Dynamics 365 Guides: Application to build AR instructional guides

Some HoloLens 2 features are provided through Microsoft's cloud computing platform, Azure. Azure Mixed Reality Services provide remote rendering and persistent real-world anchor points for virtual objects. Applications for the HoloLens 2 can be built using Unity, Unreal Engine (UE), JavaScript (through WebXR), or natively with a custom engine utilizing OpenXR. Microsoft has also published the Mixed Reality Toolkit for Unity and UE to help with development (see Section XI: AR Frameworks, Platforms, and SDKs). For the price, the HoloLens 2 starts at **$3500** for the base model, with additional charges for certain Microsoft applications and cloud services.

#### 2) MAGIC LEAP 2

Magic Leap 2 is Magic Leap's response to the HoloLens 2 and is a **standalone OST HMD**. However, the Magic Leap 2 has three major design differences compared to the HoloLens 2. First, the HMD **does not support user eyewear** like the HoloLens 2. Its form factor resembles that of traditional eyewear. Second, Magic Leap 2 implements segmented computing. The headset is tethered to an external compute pack that can be worn. The headset has a processor for low latency tasks, while intensive tasks are offloaded to the compute pack [56]. Third, Magic Leap 2 has controllers as an alternative mode of input. The Magic Leap 2 has a **resolution per eye** of $1440 \times 1760$ and an **FOV** of **44.6° horizontal** and **53.6° vertical**, to give approximately **70° diagonally** [57]. The display has a **refresh rate of 120Hz** [56]. Magic Leap recommends a minimum of 60 FPS to avoid discomfort and aiming for 120 FPS to reduce motion sickness [58]. Magic Leap 2 also comes with **Dynamic Dimming Technology**$^{TM}$. The display can remove up to 99.7% of the ambient light while retaining the brightness of the virtual content. It also supports **Segmented Dimming**$^{TM}$, where only select parts of the display are dimmed. **Ambient light sensors** on the headset support adjusting the dimming and brightness [56]. To our knowledge, this is the only commercial OST HMD that has implemented a solution for the transparency problem with OST displays.

Concerning sensors, the HMD has a **ToF depth sensor** with a **resolution of 544 × 480** and an **FOV** of **75° horizontal** and **70° vertical**. As with the Hololens, this allows the Magic Leap 2 to create spatial maps of the environment [59]. There are **3 Wide FOV "world" cameras** [56] that assist in determining the head pose, refining the spatial map, and identifying hand gestures for hand tracking [60]. The HMD is capable of **hand tracking at 30 FPS** and with **26 keypoints tracked per hand** [56]. Technically, the API lists 28 keypoints per hand [61], but the API seems to ignore two keypoints (Wrist Ulnar and Wrist Radial) in its calculation [62]. Real-time **eye tracking** is provided at **90 FPS** by 4 Eye Tracking cameras **(2 cameras per eye)** [56]. Iris recognition (Iris ID) for authentication is enabled only for certain licenses [63]. Finally, the Magic Leap 2 has a **12.6MP RGB camera** capable of **4k video at 30FPS** or **1920 × 1080 at 60FPS**. For audio, it has **2 built-in stereo speakers** and a **4 microphone array** [56]. Voice commands can be processed locally on the device [60].

For the CPU, the Magic Leap 2 sports an **AMD Zen 2 Quad Core x86 processor** (8 threads) and a custom CVIP (Computer Vision) Engine with 14 cores. It also has a GPU based on AMD's RDNA 2 GPU architecture. It has **16GB of DRAM** and **256GB of storage**. The Qualcomm FastConnect 6900 System provides wireless connectivity [64] and supports **Wi-Fi 6** and **Bluetooth 5.0** [56]. Under active use, it has a **maximum battery life of 3.5 hours**. All these components for compute and networking are placed within the compute pack [64]. The compute pack is a compromise between mobile and remote computing. It allows the processing to take place outside the HMD, allowing for a smaller and lighter HMD. The **HMD** itself only **weighs 260g** [56], [64], 296g lighter than a HoloLens 2. This design also allows for higher compute specs, freed from the very constraining form factor of an HMD. However, the limitations of being a mobile platform are still reflected in the battery life.

The headset is single size [56] but comes with a kit of assorted nose and forehead pads to customize the fit [64]. An optional overhead strap is included for comfort over long periods of use. The Compute Pack can be clipped onto waistbands or pockets or be carried using the included shoulder strap [64]. As the headset **cannot support other user eyewear**, there is an option for swappable prescription lens inserts (sold separately). The supported optical power for this insert is +5 to -10 Diopter. Magic Leap also provides **a hand-held controller** with optical tracking, IMU, and infrared to enable 6DOF tracking of the controller. Additional input devices can be supported over Bluetooth [56].

The headset runs on Magic Leap OS, based on the Android OS [56]. The Magic Leap SDK allows application development with Unity or natively (with Android Studio) [65]. It also supports OpenXR [66] and WebXR [67]. An SDK is available for Unreal Engine 5, but it is currently in preview [68]. Magic Leap also provides support to integrate third-party tools like MRTK into development [69]. Applications can be developed without the headset using the Magic Leap App Simulator [70]. Magic Leap also provides ARCloud, a remote computing service for large-scale spatial data. This service can be deployed locally or with the

supported list of Cloud Service Providers [71]. The Magic Leap 2 is available for **$3299**, with developer and enterprise editions priced **$4099** and **$4999** respectively. All models have the same hardware configuration. The difference lies in licensing and additional services from Magic Leap [63].

#### 3) META QUEST PRO

The Meta Quest Pro is a **standalone VST HMD**. It is intended to be a VR headset that supports MR with a passthrough display. Meta first introduced video passthrough for MR with the Passthrough API for the Meta Quest 2 [72]. However, the Quest 2 was limited to grayscale passthrough as it was implemented through the tracking cameras [73]. The Quest Pro improves on this by offering full-color stereoscopic passthrough [74]. The display has 2 LCD panels with local dimming backlight and a **resolution of** 1800 × 1920 **per eye**. It offers an **FOV of 106º horizontal and 96º vertical** [74]. The **refresh rate** is not listed in the tech specs, but sources claim a rate of **90Hz** [73]. A detailed list of sensors is not provided, but there are ten in total. Five thereof are infrared sensors for eye and face tracking with a 120° field of view each. The remaining five sensors support **6DOF inside-out SLAM tracking** and mixed reality [74]. These inside-out tracking sensors can also perform **hand tracking** [75]. For audio, there are **4 integrated speakers with spatial audio support**, a **3 microphone array**, and **two 3.5mm audio jacks (left/right)** [74].

The Quest Pro is powered by the **Snapdragon XR2+ platform** for the compute. It has **12GB of DRAM** and **256GB of storage**. The headset supports **Wi-Fi 6E** and **Bluetooth 5.2** for wireless connectivity. It has an **average battery life of 2.5 hours** under general use. The headset weighs **722g** [74]. The headset is single-size but adjustable. The headset can support user eyewear, as the distance between the eyes and the lenses can be adjusted. The distance between the lenses, the **(IPD)**, can be **adjusted from 55-75mm** [74], [76]. There are also partial light blockers that can be attached to reduce ambient light since this headset has a VST display [76]. A full light blocker can be bought separately [77]. The headset also comes with **two controllers for input**. Each controller has a Qualcomm Snapdragon 662 processor and three camera sensors. The three sensors help with controller tracking through SLAM [74].

The headset OS is based on Android [73]. Applications can be developed using game engines like Unity and Unreal Engine. Browser-based applications can be built using WebXR or Progressive Web Apps (PWAs) for 2D applications. Applications can also be developed natively, with SDKs provided for Mobile and PC [78]. The Meta Quest Pro is available starting at **$999.99** [74].

#### 4) META QUEST 3

Meta Quest 3 is the successor to the Meta Quest 2 and a consumer-oriented derivation of the Meta Quest Pro. It is a **standalone VST HMD** [79]. It has a **resolution of** 2064 × 2208 **pixels per eye** with an **FOV of 110° horizontal and 96° vertical**. The possible refresh rates are **72Hz, 80Hz, 90Hz, or 120Hz** [79], [80]. The passthrough for AR is provided by 2 RGB cameras and a depth projector. **Hand tracking** is supported by 4 IR cameras and the two RGB cameras mentioned earlier [79]. The Quest 3 **does not support eye or face tracking** like the Quest Pro [80]. For audio, the Quest 3 has **integrated stereo speakers with spatial audio** with **a 3.5mm audio jack** [79]. The configuration of the microphone is unknown as it is not listed in official sources.

For the compute, the Quest 3 is powered by the **Snapdragon XR2 Gen 2** SoC. It has **8GB of DRAM** and **128GB or 512GB of storage** depending on the variant [79], [80]. For connectivity, **Wi-Fi 6E** and Bluetooth are available [79]. Bluetooth is not explicitly listed in the specifications but is mentioned in battery life. The Qualcomm SoC is capable of both Bluetooth 5.2 and 5.3 [81]. We assume it to be **Bluetooth 5.2** to be aligned with the Quest Pro. Meta claims a **battery life of up to 2.2 hours** of usage on average. Weighing **515g**, the Quest 3 is lighter than the Quest Pro. The headset is single-size but adjustable [79]. While the Quest 3 can be used with eyewear, it is possible to get prescription lenses for the HMD [82]. The **IPD** can be **adjusted from 53-75mm**. The HMD also comes with **2 haptic-enabled controllers** for input [79].

The software stack and development for the Quest 3 is similar to the Quest Pro. Applications can be built with Unity, Unreal Engine, OpenXR, or WebXR [83]. The Quest 3 is priced at **$499.99 for 128GB** and **$649.99 for 512GB** [79].

#### 5) HTC VIVE XR ELITE

The VIVE XR Elite by HTC is a **standalone** XR headset with a **VST** display. The display has a **resolution of** 1920 × 1920 **per eye** (3840 × 1920 combined) with a **refresh rate of 90Hz**. It claims an **FOV of up to 110°**, but no information is specified on the horizontal, vertical, or diagonal. The headset has 6DOF inside-out tracking. The headset has a **G-sensor (accelerometer), gyroscope, depth sensor, proximity sensor**, and four tracking cameras [84]. The headset supports **hand tracking** [85] but has no eye or face tracking sensors. **16MP RGB camera** allows capturing the environment for video passthrough. There are **embedded speakers** and **dual-integrated microphones** for audio. Compute is provided by the **Qualcomm Snapdragon XR2 platform**. It has **128GB of storage** and **12GB of memory**. For wireless connectivity, it supports **Wifi 6E** and **Bluetooth 5.2**. It has **2 USB 3.2 Gen-1 Type-C ports**, one for power and one for peripheral. It has a 24.32Wh battery cradle that is removable and hot-swappable [84] and can provide up to **2 hours of continuous power** [85].

The headset is single-size but adjustable. A feature of this headset is that the battery cradle can be removed to change the form factor to one more akin to that of normal eyewear. The diopters of each lens can be adjusted and allow the user to use the headset without prescriptive eyewear [85].

The **IPD** can be adjusted within a range of **54-73mm** [84]. It should be noted that the battery cradle is claimed to act as a counterweight to the display [84], [85]. The long-term comfort of the headset may be affected if used without the battery, but the headset only weighs **273g**. It weighs **625g** with the battery cradle headstrap [86]. An alternative power source must also be provided if the battery cradle is detached, with at least 30W of power delivery [85]. Included with the headset are **2 controllers** with up to 15 hours of battery life. However, the controller's tracking is provided by a G-sensor and gryoscope [84], unlike the optical tracking seen in other systems.

The headset runs on Android [86]. HTC provides the VIVE Wave SDK for application development. The SDK is available for Unity, Unreal, and Native (Android) development [87]. Applications can also be developed with OpenXR [88] as the device is conformant to this standard [89]. The headset is available for a retail price of **$1099** [86].

#### 6) VARJO XR-3

The Varjo XR-3 is a **tethered VST HMD**. This HMD's display is a key feature as it claims human-eye resolution [90]. This is achieved by taking advantage of the nature of the human eye. The resolution of the human eye is not uniform across the visual field in a given instance. The human eye's resolution can be matched by providing a high resolution for the fovea and a low resolution for the periphery. This rendering technique is called **foveated rendering** [31]. The XR-3 display has a **focus area** uOLED with a **resolution of** 1920 × 1920 **per eye**. It has an FOV of 27° x 27°, giving it roughly 70 Pixels Per Degree (PPD). The **peripheral area** has a 30PPD LCD with a **resolution of** 2880 × 2720 **per eye**. The display overall has a **horizontal FOV of 115°** and a **refresh rate of 90Hz**. It also supports a wide color gamut, supporting 99% of sRGB and 93% of DCI-P3. The foveated rendering is enabled by the **built-in eye tracking**, operating at 200Hz with sub-degree accuracy. This allows the headset to show the user what they directly look at in full resolution [90].

Positional tracking is provided by utilizing the RGB video pass-through cameras for inside-out tracking. There are two 12MP cameras with a latency of $<$ 20ms for passthrough. Alternatively, base stations can provide positional tracking (sold separately). Depth sensing is provided by a fusion of **LiDAR** with the stereo RGB feed, with an operating range of 40cm-5m. **Hand tracking** is provided by **Ultraleap's Gemini v5 software** [90], [91]. There is **no in-built audio**, but there is **a 3.5mm audio jack with microphone support** [90]. Since this is a tethered headset, the processing must be performed on an external computer. The XR-3 can, therefore, utilize/require much more computing power to operate compared to its competitors [92]. The headset is single-size but adjustable and does support user eyewear. It also has **automatic IPD adjustment** with a range of **59-71mm**. It weighs **980g**, with the **headband alone weighing 386g** [90].

Since this tethered HMD outsources processing, it has no OS. Applications can be developed using Unity, Unreal, and OpenXR [90], [93]. There is support for a broad range of professional 3D software [93]. Native development is possible through the Varjo Native SDK [94]. Varjo also offers various services and applications of their own, namely:

- Varjo Base: Companion software for headset configuration and monitoring [95]
- Varjo Workspace: Virtual Windows Desktop for Windows Applications [96]
- Reality Cloud: Remote processing, session management, and shared AR sessions [97]

The Varjo XR-3 is **available for €6495**. However, the headset cannot be purchased without buying at least a 1-year Varjo subscription. It costs €1495 for the mandatory 1-year subscription, raising the **total price to €7990**. The subscription features include software updates, commercial licenses for Varjo and Ultraleap software, access to a Varjo Account portal, and technical support for businesses [98].

#### 7) VRGINEERS XTAL 3 MIXED REALITY

Vrgineers is a company that specializes in pilot training systems for both professional and military clients. Their latest offering, the XTAL 3, was developed to be a pilot-dedicated headset [99]. However, it is not limited to this thanks to its wide support for developer software [100], [101]. It is a tethered headset with both a VR and MR variant. The central cover of the headset is removable and reconfigurable. This makes it possible to configure the VR headset to support MR [99]. This is done with the addition of RGB cameras for passthrough. Thus, it is a **tethered VST HMD**. The two key features of this headset are the wide FOV and the high display resolution. Vrgineers claim an **FOV of 180° horizontal and 90° vertical**. However, the actual FOV for MR can vary depending on the lens used. On the lower end, it has an **FOV of 63° horizontal and 38° vertical with a high-fidelity lens** with 60 PPD. On the higher end, it can offer **175° horizontal and 100° vertical with the Infinite lens**, at the cost of the PPD being lowered to 22 [101]. The HMD has one LCD per eye. For MR, it can provide a resolution of **3864×2192 at 45Hz or 2232×2192 at 75Hz per eye**. By **utilizing foveated rendering**, it is capable of **3840×2192 at 90Hz per eye**. To support foveated rendering, **eye tracking** is provided, running natively at 120 Hz (up to 210 Hz) [100], [101].

Positional tracking is provided by inside-out tracking. It can also support other 3rd party tracking solutions such as Optitrack or Vicon [99], [100]. **Hand tracking** is provided by the **embedded sensor from Ultraleap** [99], [101]. The headset also features **automatic IPD adjustment**, with a range of **60-76mm** [100]. The wide FOV of the HMD is not without its issues. To compensate for the distortion, Vrgineers use a combination of their embedded lenses and warping algorithms [99]. The wide FOV also makes the optical system focus on the standard reading distance [100]. This can be a problem for users with eyewear. Vrgineers offers a service to

prepare custom corrective lenses with every order [99], [100]. The headset is single-size but adjustable. Using user eyewear may be possible but will very likely be uncomfortable. The headset is heavier compared to competitors, weighing **700g without including the headstrap**. Since it is a tethered headset, it connects to a PC with **DisplayPort 1.4 and USB 3.2 Gen 2** [100], [101].

Since this is a tethered headset, it has no OS. Microsoft Windows is the only supported OS for the tethered PC. The headset supports a wide range of flight simulators, keeping with its origin as a pilot-dedicated headset. Other applications can be developed through Unity and Unreal Engine, and it also supports SteamVR and OpenXR [100], [101]. Vrgineers also provides the VRG C++ API for native application development with C++ libraries [102]. The XTAL 3 is available with a starting price of **$11800** for a personal free license [100]. There are also plans to release a wireless version of this headset. The headset will not be standalone owing to the lack of onboard computing. However, the PC can stream data to a separate on-belt module containing the battery pack and wireless antennas [103].

#### 8) AR GLASSES

AR glasses are a category of devices in the industry that attempt to constrain the HMD to the small form factor and weight of regular eyewear. These constraints limit them in display, input, and compute capabilities. However, as technologies improve, the feature-rich HMDs covered above may shrink to the size of AR glasses. It is important to note that we use AR glasses as a subjective classification and are not based on an objective definition. To better gain an idea of the devices in this class, a selection of five devices have been chosen:

- Google Glass Enterprise Edition 2
- Epson Moverio BT-40S
- Nreal Light
- Nreal Air
- Vuzix M400

The specifications for each device have been provided in Table 11 (see Appendix). These devices have been chosen to convey this class's variety rather than showcase the latest commercial solutions. The differences primarily manifest in the device type, display, and registration capabilities.

### C. SPATIAL AR (SAR)

The field of SAR has not enjoyed the same attention as HMDs or smartphones in the industry. No complete solutions could be found for sale at the time of writing. One possible reason would be the challenges of SAR (see Section IV-B3: Spatial AR under Enabling Technologies: Display). These limit the scenarios in which SAR can be deployed. This can reduce the demand needed to drive mass adoption and development. At the time of writing, Lightform was the only provider of complete SAR solutions that could be found. Unfortunately, the company has shut down, with all products discontinued and services stopped [104]. Nevertheless, their products are still worth covering as examples.

LF2+ was a complete SAR solution from Lightform. The unit consisted of the following [105]:

- a projector for the display
- a camera array for scanning and estimating depth
- compute for processing
- onboard storage
- an integrated microphone (input)

The LFC kit was another product by Lightform for adding SAR functionality to almost any projector. The kit consisted of the Lightform Compute (LFC) unit, a Logitech Brio webcam, and a projector mount. The projector mount could mount the webcam and LFC unit to a projector [106]. Table 1 provides the full list of specifications for both products. The Lightform Creator app was used for developing experiences and was provided for free with the LFC kit or LF2+ [105], [106], [107]. The app simplified the process of projection mapping and the implementation of audio reactivity [107].

Audio reactivity is the most developed and integrated form of interaction for Lightform products. However, audio reactivity is simply the system reacting to the audio captured from the surroundings, such as a sound-reactive visual effect [108]. While it is a form of interaction, it is not particularly meaningful beyond being a gimmick. There are other means of interaction, but over the Open Sound Control (OSC) protocol. OSC is a protocol originally intended for networking between sound synthesizers, musical instruments, and other multimedia devices [109]. This can be used to make interactive controllers with phones through the TouchOSC app [110]. It can also be used to interface with Microsoft Kinect sensors. The sensors could be used to incorporate body tracking, but only one example is provided. Other sensors with OSC libraries for communication could also be utilized [111]. However, these OSC expansions were experimental [110], [111]. They were not well developed, providing only limited interactions out of the box. While Lightform products are technically complete solutions, they are practically incomplete.

Lightform was aware of this, and their plans can be seen with Project LFX [112]. The foundation of this system would be projectors capable of adjusting their orientation to project content anywhere in a given space. This is meant to be combined with more unconstrained modes of interaction, like voice commands. The system's potential can be seen with their stated design principles [113], and it even highlights some of the advantages of SAR.

While there may be no complete and commercial solutions, there are still commercial solutions that could be combined for SAR. An AR system can be built if it possesses all three enabling technologies. Software for projection mapping is commercially available. They are intended to be used with almost any projector on the market. Such projection mapping software may also come with a means for interaction. For example, MadMapper comes with support

for MIDI/OSC-based controllers [114]. This can be expanded to communicate with other sensors, similar to Lightform. Sensors like Azure Kinect for 3D sensing have already been made to empower XR applications. This can add body tracking, speech recognition, and other computer vision capabilities to a system [115]. These sensors can be used to enable tracking, registration, and input. If such sensors cannot be obtained, computer vision techniques could be utilized with any camera to implement similar functionality. Table 2 shows a list of projection-mapping software and 3D sensing sensors that could be used for custom SAR solutions.

### D. OTHER POSSIBILITIES

This section discusses planned, unreleased, and unconventional hardware implementations.

#### 1) MICROSOFT HOLOLENS 3

Microsoft has hinted plans to update the HoloLens 2, but no more information has been revealed. The rumored HoloLens 3 is suffering from issues with the development team, and thus, its future remains uncertain [116].

#### 2) APPLE VISION PRO

The Vision Pro is a **standalone VST HMD** from Apple, scheduled for release in early 2024 in the U.S. with a price of **$3,499** [117]. Apple claims a resolution beyond 4K with two micro-OLED displays having 23 million pixels between them [118]. For reference, the standard 4K resolution at $3840 \times 2160$ has about 8.3 million pixels. The FOV is unknown. The **LiDAR scanner** and **camera array** can scan and construct a 3D map of the user's environment. **Hand tracking** is supported, with infrared flood illuminators to support hand tracking in low light conditions. **Eye tracking** is supported through a system of infrared cameras and LEDs. It has **integrated speakers supporting spatial audio** and a **microphone** [118].

For computing, the Vision Pro comes with Apple's proprietary **M2 chip** for running the OS and applications and an **R1 chip** dedicated to dealing with sensor input and streaming images to the displays. The HMD is single-size but adjustable with a flexible headband and a dial to adjust the fit. Compatibility with eyewear is unknown, but prescription inserts can be obtained to use the HMD without eyewear. The HMD has a split design like the Magic Leap 2 with an external battery pack. This provides up to **2 hours of battery life**. The HMD can also be used plugged in, allowing all-day usage [118]. It is unknown whether the Vision Pro can utilize external computing in such a configuration since the user will be tethered. This would depend on the connection used.

Apple is heavily betting on software and integrated features for the success of this device, powered by their **VisionOS**. The HMD has a feature called **"EyeSight"**, a display at the front of the HMD that allows other people to maintain eye contact with the user. It can also enable other users to know how immersed the user is in an experience (along the virtuality-continuum). VisionOS will simultaneously display a user's eyes and make people in the environment visible when they approach the user. The microphone is integrated with the OS for dictation. Eye tracking is integrated as a mode of input, specifically for intent. Apple has revealed many intended applications for their HMD. Remote displays will allow the user to carry a computer's display anywhere. The HMD or an iPhone can capture spatial videos as 3D recordings with depth, allowing users to relive experiences as if they are in them. Apple is also promising integrations for collaborative video conferencing that can take place spatially [118].

#### 3) HolokitX

The Holokit X by Holo Interactive is an accessory that can convert an iPhone into a stereoscopic AR display [119]. The idea of converting a smartphone into a headset is not new. For instance, Cardboard by Google would allow converting smartphones into stereoscopic VR headsets [120]. It is easy to think that video passthrough from the phone's camera is all that is needed to convert such a VR HMD to a VST HMD for AR. However, the outward-facing cameras on a smartphone are kept together and towards one side of the phone. Thus, the real-world view of the cameras would never be aligned with what the user would normally see (see Section IV-B4: OST vs VST HMDs under Enabling Technologies: Display). The Holokit X solves this by utilizing an OST display instead, offering an FOV of 60°. The digital objects are projected from the phone display to the user. The headset is single-size but adjustable and is compatible with user eyewear. The iPhone and Apple's AR platform (ARKit) provide AR functionality. The system is capable of 6DOF tracking, environment scanning with LiDAR, human and object recognition, visual hand tracking, and hands-free motion control (with Apple Watch). It also offers strong multi-user functionality. Additionally, it provides a spectator view, where users with iPhones or iPads can view and record the AR session experienced by other Holokit users in third-person [119]. The Holokit Unity SDK allows applications to be developed with Unity and is compatible with ARFoundation by Unity [121]. The Holokit X is available for $129 [119]. However, since the functionality depends on an iPhone and other Apple devices, the price can easily balloon into the range of dedicated MR HMDs. As an alternative, there exists an open-source Holokit project, similar to Google Cardboard, to turn smartphones into OST AR HMDs. While it can support Android and iOS devices, the compatible list of devices is limited [122].

#### 4) ZAPBOX

Zapbox by Zappar is a kit to convert iPhones into AR HMDs [123], similar to the HolokitX. Zappar is a provider of AR solutions [124]. Unlike the HolokitX, the Zapbox converts an iPhone into a VST display. Zapbox also provides

**TABLE 1.** Specifications of Lightform products for SAR.

| Name | LF2+ | LFC Kit |
|---|---|---|
| **Resolution** | 1920 x 1080 (FHD) | 1920×1200 (WUXGA)<br>4K projectors support (with upscaling) |
| **Brightness** | 1000 Lumens | 1000-100K Lumens |
| **Throw Ratio** | 1.2 : 1 | 0.5 – 2.0 |
| **Scan and Project Range** | 1.0-3.0m distance<br>(corresponds to 1.2-3.6m scene width) | - |
| **Digital Keystone** | Not supported / Disabled | - |
| **Dynamic Contrast Ratio** | 10000 : 1 | - |
| **Lamp Life** | 30,000 Hours | - |
| **Processor** | 1.8GHz Quad core Arm Cortex-A53 | Intel Celeron N4000 (Gemini Lake)<br>8GB DDR4 RAM |
| **GPU** | Arm Mali-T830MP2 | Intel UHD Graphics 600 (iGPU) |
| **Camera** | 5MP RGB camera<br>IR camera | 13MP RGB camera<br>IR sensor |
| **Storage** | 20 GB eMMC | 32 GB eMMC |
| **WiFi** | 802.11 ac/b/g/n (2.4GHz / 5GHz) | 2.4 GHz & 5 GHz |
| **Connectivity** | LAN<br>HDMI x2 | LAN<br>HDMI x1 |
| **Microphone** | Integrated microphone | Stereo mic |
| **Extra non-functioning ports** | USB 2.0 x1<br>USB 3.0 x1<br>Headphone Audio Out<br>SPDIF Audio Out | USB 3.0 x4<br>HDMI x1<br>LAN x1<br>Audio out<br>MicroSD |
| **Dimensions** | 8.26 in (L)<br>7.87 in (W)<br>4.13 in (H) | 4.6 in (L)<br>4.6 in (W)<br>1.3 in (H) |
| **Weight** | 5.1 lbs | 0.63 lbs |
| **Power** | 19Vdc/6.32A (120W) | 12Vdc /3.0A (36W) |
| **Noise** | <40dB typical (50dB max) | N/A |

**TABLE 2.** A list of projection-mapping software and 3D sensing sensors for custom SAR solutions.

| Projection-Mapping Software | 3D Sensing/Tracking Sensors |
|---|---|
| MadMapper | Azure Kinect |
| HeavyM | Kinect V2 [1] |
| Resolume Arena | Kinect V1 [2] |
| Isadora | Intel Realsense D435 |
| TouchDesigner | Ultraleap Leap Motion 2 |
| Mapping Matter | Ultraleap 3Di |
| MapMap | Asus Xtion PRO LIVE |
| FaçadeSignage | Primesense Carmine 1.09 |
| VPT | Orbbec Astra Pro Plus |

[1] Same as Xbox One Kinect Sensor. Use Kinect SDK v2
[2] Same as Xbox 360 Kinect Sensor. Use Kinect SDK v1

two Bluetooth controllers for input [123], and a Unity SDK for development [125]. It is priced at $79.99 [123].

#### 5) MONOCLE

Another interesting solution is the Monocle by Brilliant Labs. The Monocle is, as the name suggests, a monocle but with built-in electronics. It is a device that can be clipped on, converting any traditional eyewear into an OST HMD. It comes with a 720p camera and a micro OLED display with a resolution of 640 × 400. The optics can project the display to the eye with an FOV of 20° [126]. A microphone and touch buttons are provided for input, and Bluetooth 5.2 is used for connectivity. The Monocle also has an FPGA for accelerating ML/CV tasks [127]. It weighs just 15g and has a battery life of up to an hour. An accompanying charging case has the capacity for six recharges. It is available for purchase at $349 [126]. It supports open source development, and applications can be developed using Python (running an OS based on MicroPython) [127].

#### 6) MOJO LENS

The final device we cover is the Mojo Lens by Mojo Vision. The Mojo Lens shrinks the hardware for AR into a form factor of a contact lens. It has a 0.48mm MicroLED display, an IMU, an ARM processor, a 5GHz radio, micro-batteries, and a circuit for wireless recharging and power management. The display is opaque and placed in the middle of the eye. However, its small size and proximity effectively render it invisible. The user can only see the image that it projects. The device has a very small FOV of 15°. However, by placing the device directly on the eye, the display can move with the eye. This alleviates the FOV limitation to some extent. It can also be considered as a form of foveated rendering. Gaze tracking is intended to be the primary (and probably only) mode of input with this device [128]. The project was still under development when it was revealed. However, the company has decided to pivot into other ventures, citing unfavorable market conditions for continued development of the Mojo Lens [129].

## VII. AR SOFTWARE

Hardware means nothing without adequate software to back it up. Devices with well-developed software frameworks and tools for development will almost always win out against those without. Taking care of the foundational elements of

an AR session, such as registration and display, frees up the developer to focus on the application itself. Choosing the right software platform can reduce development times and ease deployment and management. Good platforms will also attract a large community, further increasing the support available to developers. Refer to Fig. 4 for a visual breakdown of all the sections under the topic of ''AR Software''. For easier referencing, the features of AR software will be highlighted in its discussion if present. Furthermore, the naming and definition of features are not consistent between software. If the feature name or functionality in the software does not match the scheme in our paper, it will be mapped with the following format: **Feature Name in Software (Feature Name in our Paper)**. Features not in our feature list are left as is.

Each section has been provided a preamble for their discussions. In the upcoming sections on Basic and Advanced Features that can be expected from AR software, we have provided illustrations to showcase the features with a dummy application. The application showcases ''THE CUBE''. The floating cube and its wooden base (where applicable) have been used to show AR content placed in the real world. These illustrations were fully rendered in UE5 (see Fig. 5). We used the following resources and tools to assemble and create our renders:

- Archviz Interior: A UE4 template project; It was used for the interior in our renders.
- Quixel Megascans: A collection of scanned 3D assets. Assets from this collection were used to improve the interior's photorealism.
- MetaHuman Creator: This tool can used to create photorealistic humans that can be imported into UE5.
- Cesium: A tool for 3D geospatial applications; It was used to create the world in Fig. 13.
- Blender: A Digital Content Creation (DCC) tool for 3D. This was used to create any custom 3D assets.
- Krita: A DCC tool for 2D. This was used to edit the images.

## VIII. BASIC FEATURES

To make it easy for future reference and avoid redundant information, we decided it was best to discuss a list of AR software features first. Such a list would also make it easy to compare between various software. We have classified these features into two groups: basic and advanced. We have selected the features in this section to be basic based on their commonality. Most, if not all, AR software will have at least one of the basic features we have classified. Being registered in 3D is a characteristic of AR. The features classified here as basic assist with enabling this particular characteristic. Thus, these features form the foundation of most AR software.

### A. POSITIONAL TRACKING

Positional tracking means that the system can estimate the device's position and rotation in the environment. This allows users to both look around and move around in the scene. There are two types of tracking available for AR devices: 3DOF and 6DOF tracking [130]. Most AR solutions provide 6DOF tracking under the name ''positional tracking''. With 3DOF tracking, it is typically restricted to only tracking rotation along three axes: pitch, yaw, and roll [130]. The user will only be able to look around with the device. This means that the AR content will be anchored to the device and will move with the device's position. With 6DOF tracking, it can track three axes each for rotation and position [130]. Virtual objects will no longer move around with the device (unless forced to).

### B. FEATURE POINTS AND POINT CLOUDS

Most tools rely on identifying distinct features in the visual feed to support positional tracking. These ''feature points'' are then used to compute location changes [131]. Since these feature points correspond with the environment, they can also be used to place AR content into the world (see Section VIII-D: Hit-testing). It may also be possible to obtain a collection of feature points as a point cloud may also be possible. A point cloud is a 3D representation of a collection of points in space.

### C. PLANE TRACKING

Plane tracking can be used to detect and track planar surfaces in the scene. Finding a planar surface typically involves identifying feature points on the same planes [131], [132]. The basic functionality of this feature is to detect horizontal surfaces facing up, like floors or tables. Support for horizontal surfaces facing down (e.g., ceilings) and vertical surfaces (e.g., walls) are also becoming more common. The functionality can also be expanded to other arbitrary planes, such as ramps in certain libraries.

Once the planes have been detected, virtual objects can be placed on them (see Section VIII-D: Hit-testing). Physics can also be applied to the virtual objects and planes. This can make it seem like the virtual objects interact with real-world surfaces. A caveat, however, is that the illusion of physics can break easily once the virtual object goes beyond the bounds of a tracked plane. Additional care should be taken when positioning an object on a plane. The lack of occlusion can easily break the illusion of a place object from another view or position (see Section IX-L: Occlusion).

Fig. 6 showcases an application for Plane Tracking. By tracking horizontal planes, the system can detect flat surfaces onto which AR content (black cube with a wooden base) can be placed. It is not uncommon for Plane Tracking implementations to track multiple planes in the same region. Fig. 6 showcases three tracked planes in translucent orange. Notice that the two planes underneath the AR content appear in the same region and appear to overlap. Additionally, notice the lighting on the AR content. The real world is lit from the right and has shadows going to the left. The AR content is lit from the viewer's perspective, indicated by the shadow

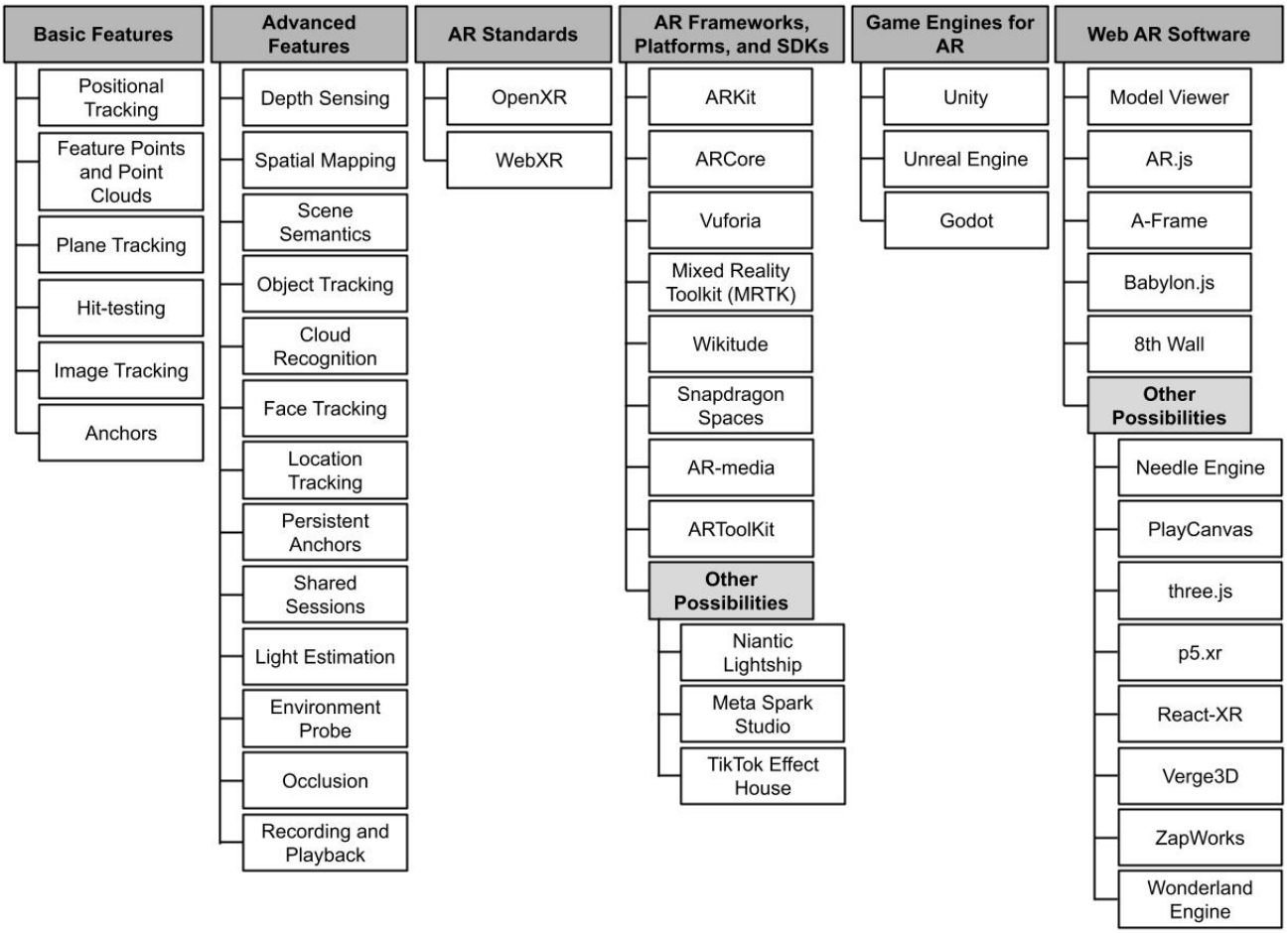


**FIGURE 4.** A visual breakdown of the section "AR Software". Primary topics are at the top (gray) and go from left to right.

going away from the viewer (on the wooden base). AR objects are rendered separately with virtual lighting and overlaid on the real image. The content looks unnatural and stands out because the lighting on the object is inconsistent with the real-world environment. This problem of having consistent lighting is addressed by Light Estimation (see Section IX-J: Light Estimation).

### D. HIT TESTING

Placing a virtual object directly onto the scene involves additional steps beyond detecting and tracking scene features. The virtual object must have the correct placement and size to blend into the scene. The problem is determining the proper point for placement in the scene. This can be resolved with hit-testing. It is also called raycasting as it follows the same principle. Raycasting is a technique used in game development. It is used to trace a line in 3D space (casting a ray) to query if any entities are being intersected by the line [133]. This should not be confused with the rendering technique of the same name. Most development tools for AR use the term hit-testing over raycasting. Performing a hit test will extend a ray from a chosen origin point. Hit-tests can return any tracked entities in the scene it intersects with. The intersection point can be used to identify the position where the virtual object should be placed. The pose of the tracked entity can be used to determine the pose of the virtual object to be placed.

The first result is typically considered if multiple results are obtained from a hit test. As the objects nearest to the origin point will be intersected first, the list of results will be sorted by increasing distance. Thus, the first result will be the closest and where we want to place the object. The purpose and behavior of a hit test can be adjusted by changing the ray's origin point, direction, and length and how the results are processed. For example, object placement has different implementations on a handheld compared to an HMD. On a handheld, the origin point can be the device camera or even a point on the screen where the user tapped. The direction of the ray can be obtained from the device's orientation. On an HMD, the origin point can be a hand-held controller, the user's hand (from hand tracking), or the HMD. The ray's direction can be obtained from the orientation of the entity chosen as the origin.

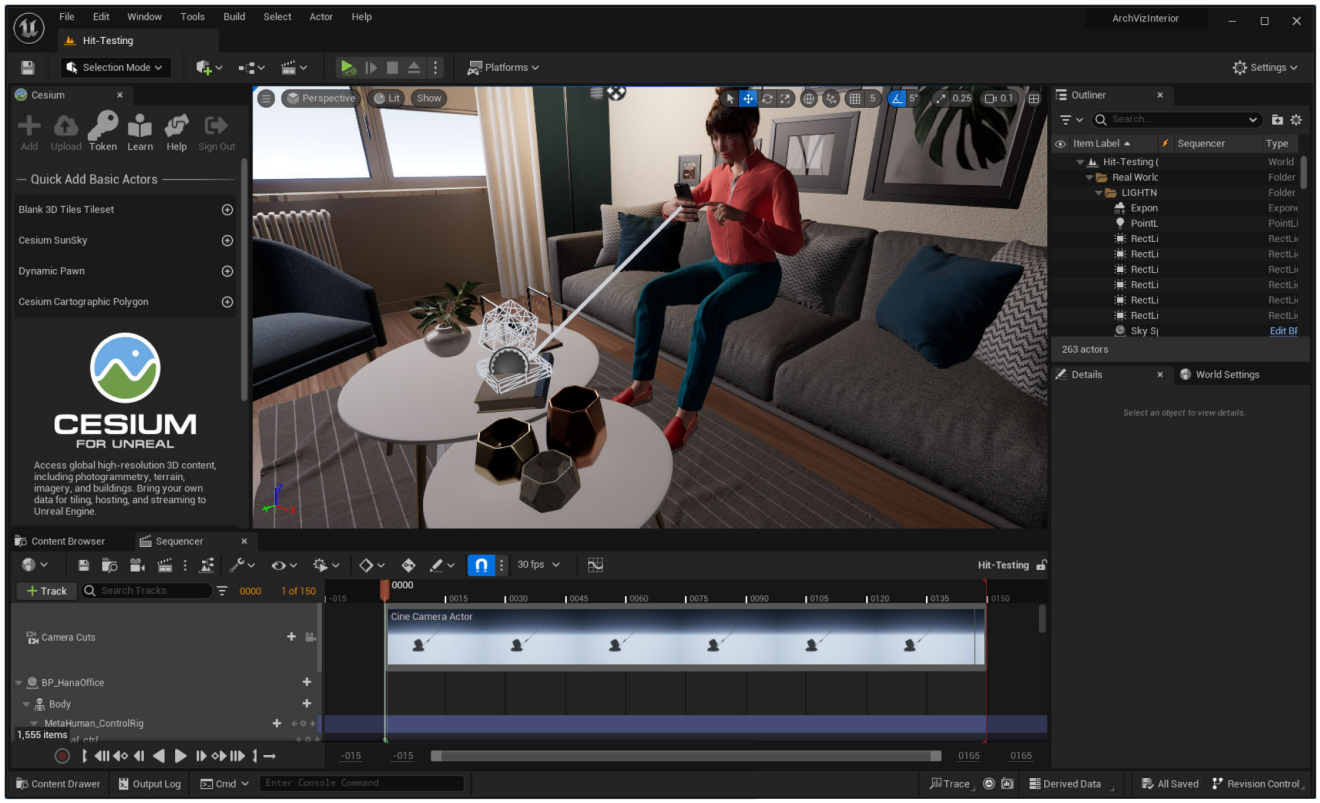


**FIGURE 5.** A screenshot of the Unreal Engine project used to create the renders in this paper. It shows a behind-the-scenes view of Fig. 7.

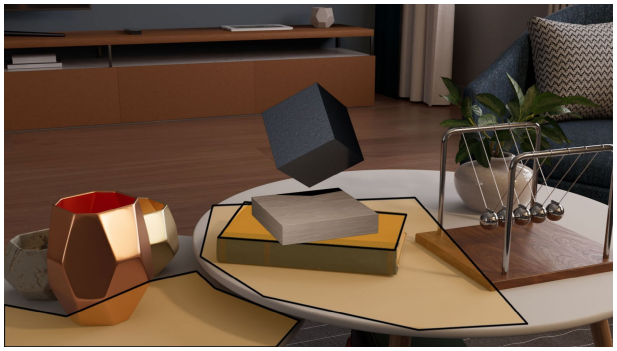

**FIGURE 6.** AR content (cube with a wooden base) placed on planes tracked in the environment (orange surfaces).

**FIGURE 7.** A visual representation of utilizing hit-testing (white line) for object placement.

Fig. 7 visually represents how hit-testing works for object placement. Here, the ray originates from the phone held by the user. The ray intercepts the plane tracked on the book shown in Fig. 6 and displays an indicator for the object to be placed. The user can then confirm the position in which to place the object.

### E. IMAGE TRACKING

Image tracking can be used to detect and track images placed in the environment. First, a library of images to be detected is created. An algorithm then attempts to detect these images from the visual feed of the world. Once detected and tracked, the images can then be used to place AR content in the scene.

The challenge with image tracking boils down to image quality. Tracking quality depends on the image's quality and its appearance in the visual feed. Computers do not see like humans. They rely on algorithms to distinguish and identify an image from its surroundings. The harder it gets to distinguish and identify, the greater the computational power required, which is a constraint for AR devices (see Section V-B: Remote Computing).

Algorithms simplify the problem to negate the need for more computing. Every image is processed into something

easier to track. An example would be reducing an image to its most important features and searching for only those features in the visual feed. However, this approach effectively limits what images can be used for tracking. Images must always conform to certain rules for optimal tracking depending on the underlying algorithm.

The most basic implementation of image tracking is (fiducial) marker tracking. These images/markers are meant to act as reference points (hence the name fiducial) and be easily recognized by a computer. Markers can be divided into two categories: barcode and pattern. Barcode markers are extremely limited in their appearance. They are bi-tonal (black and white), with variations of a set pattern. They are akin to QR codes but more simple in their appearance to aid with recognition [134].

Pattern markers are more flexible. These markers are white squares with a black bounding box. Unlike barcode markers, these markers can have their appearance customized. This can be done by placing a pattern in the white square (hence the name). There are still limits when it comes to the pattern that can be used. This is because pattern markers break down the pattern into a grid of squares [134]. Therefore, complex patterns or patterns with low contrast to the background will be difficult to track.

More modern implementations provide support for typical images over markers. They also provide support for tracking multiple images at once. However, image tracking requires more computational power than marker tracking. As such, multiple marker tracking may be preferable in certain scenarios. Some development tools also expand support for image tracking on curved surfaces, such as cylinders or cones.

Image tracking can be expensive to compute, so it is best to limit the number of images that require precise tracking [135]. For similar reasons, image tracking performance can degrade if the library of images to be detected is too large. Image libraries, therefore, usually have a hard or soft limit on the number of images that can be stored. If more images are needed, one can switch to a different set of images depending on the context, such as location [135]. Alternatively, features and facilities like Cloud Recognition can be used to support large reference libraries.

Fig. 8 shows one application of image tracking. Outside of acting as a tracked point for AR content, image tracking can make images interactive or give them depth. Here, the system can identify a poster in the environment, causing an AR cube to pop out of the wall.

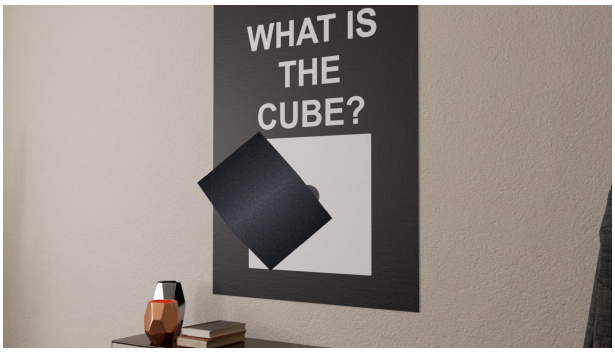


**FIGURE 8.** AR content (cube) placed next to a poster using image tracking.

### F. ANCHORS

Anchors are used to enable object permanence in AR. Losing the line of sight means losing the ability to track the scene. This, by extension, means losing the ability to display virtual content in its placed position in the scene. By attaching/anchoring virtual content to anchors, they appear to remain in place in the scene, even if they go out of sight. This is done by updating the transform of the anchor relative to the device transform every frame. Anchors can be used with tracked entities, such as planes or images, to ensure they remain tracked even when out of sight. Alternatively, they can also be used to track an arbitrary point in space. These are sometimes called Spatial Anchors. Besides permanence, anchors can be saved to memory to create persistence (see Section IX-H: Persistent Anchors) and multi-user experiences (see Section IX-I: Shared Sessions) [136].

## IX. ADVANCED FEATURES

Advanced features are typically found in more well-developed AR software. In general, they are more specialized and complex functionality beyond assisting with the third characteristic of AR. While advanced features like location tracking and object tracking do help with the third characteristic of AR, they are not as common as their basic counterparts.

One feature we have left from our list is ''Instant Placement''. It typically refers to the ability to instantly place an object into the environment (see sections XI-A(ARKit) and XI-B(ARCore) under AR Software). This is because AR systems have to wait until tracking is established and to be accurate enough to place an environment in the world. We argue this is not a feature as it provides no unique functionality. Rather, it is a measure of the system's capability to provide accurate tracking in a time small enough to be seen as instantaneous.

### A. DEPTH SENSING

Some tools provide an API for accessing the depth information of the scene. This allows a system to know the physical objects are positioned relative to each other in the world. Utilizing a dedicated hardware sensor like LiDAR can improve depth estimation accuracy. Depth information can be utilized in a variety of ways. This can be used to enable physics or occlusion (see Section IX-L: Occlusion). It can also be used to apply visual effects, such as depth of field or fog [137].

### B. SPATIAL MAPPING

Spatial mapping (or scene mapping) is a feature used to create a 3D scan of the scene. Creating a 3D model of the scene

**FIGURE 9.** Spatial mapping: The real world environment to be mapped.

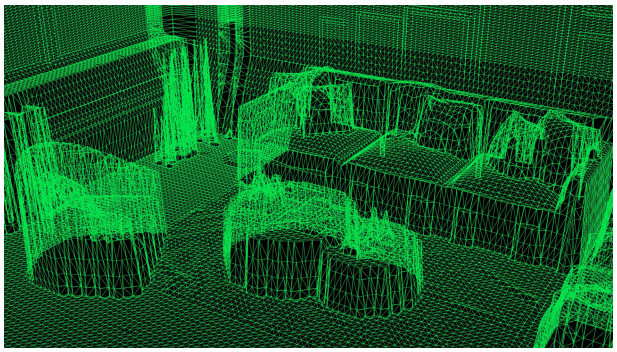

**FIGURE 10.** Spatial mapping: The mapped 3D model of the environment.

makes it possible to keep track of entire sections of the scene outside of view. This has multiple applications. Since the system has a 3D model of the environment, it can calculate physics even in spaces that cannot be seen, just like reality. The 3D model can also be utilized to enable occlusion. This can provide a great sense of realism and immersion. A 3D map of the scene can also be used to simulate the environment without needing to be there. This reduces the effort involved in iteration and testing, speeding up development.

Fig. 9 shows the real-world environment, and Fig. 10 shows the scanned 3D model. The 3D model will never be completely accurate from a scan. For instance, notice how regions below the chair, couch, and table are not fully mapped.

### C. SCENE SEMANTICS

Scene semantics is used to classify and label objects within the scene. This makes it possible for the system to distinguish a tabletop from the floor or buildings from the terrain. Such classification will allow the creation of more adaptable and tailored AR experiences. Scene semantics must not be confused with spatial mapping. That being said, scene semantics can be used in conjunction with spatial mapping to create a labeled 3D model of a scene.

### D. OBJECT TRACKING

Object tracking can be used to track physical objects in the scene, which can then be used to place AR content into the


**FIGURE 11.** AR content (text and target marker) placed with object tracking.

world. A 3D model of the object is required for detecting and tracking the object. Some developer tools provide the ability to scan a physical object to produce a 3D model with a process called 3D scanning. In some cases, this can also be done with the AR device itself by leveraging the same techniques it uses to map and understand the scene.

Fig. 11 showcases an application for Object Tracking. Objects can be identified in the environment and marked, and tooltips can be placed next to them. In Fig. 11, the system looks for a regular dodecahedron in the environment after being provided with its 3D model. It then places a marker with text to alert the user it has discovered the object.

### E. CLOUD RECOGNITION

With certain types of tracking, such as image tracking, it is necessary to keep a library of references. However, this can pose problems in scenarios requiring a large reference library. The system will need to search for multiple images in the feed and store a large database of images. Mobile hardware will struggle as the references start to number in the thousands. Cloud recognition aims to resolve this dilemma. It removes the need for processing the visual feed and storing the reference library locally on a device. Instead, the reference library and visual feed can be sent to the cloud for storage and processing respectively.

This has both benefits and drawbacks. The capabilities of mobile hardware are improved with remote computing (see Section V-B: Remote Computing). The cloud has vastly more compute resources than mobile hardware and is scalable. As the reference library is stored and accessed in the cloud, it is easier to maintain the database, as all end-point devices share it. However, a major concern would be privacy. Cloud recognition is mostly provided for image recognition with 3rd party cloud service providers. The visual feed is being sent to a 3rd party for processing. Thus, a 3rd party can have access to possible sensitive information.

### F. FACE TRACKING

Face tracking is used to track a user's face and facial expressions. With appropriate processing, it can be utilized as a supplementary method of input. This can be utilized in

**FIGURE 12.** Placing AR content (glasses) with face tracking.

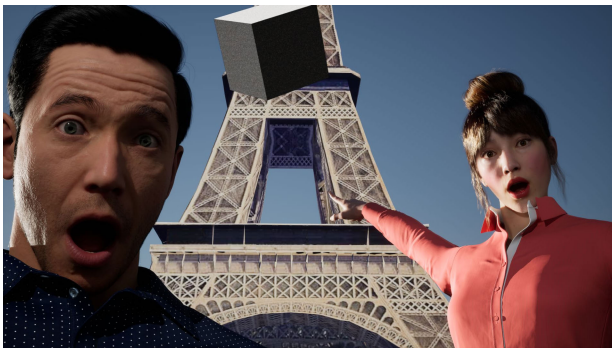

**FIGURE 13.** Location tracking can be used to place AR content (cube) in real-world locations.

a variety of ways. It can be used for motion capture, which can be used to create animations, or used in real-time for animating user avatars. This could be used to resolve some of the social challenges arising from using HMDs, particularly VST HMDs (see Section IV-B4: OST vs VST HMDs under Enabling Technologies: Display). It can also be a foundation for applying visual effects to the face. This could used to preview eyewear or makeup on the face, as shown in Fig. 12. By tracking the user's face, it is possible to preview eyewear by positioning the virtual content correctly around the eyes. Notice that the lighting of the eyewear in Fig. 12 is flat and inconsistent with that of the real world.

### G. LOCATION TRACKING

Location Tracking can be used to deliver location-specific AR experiences. The basic implementation relies only on GPS data to determine where the user is in the world. AR content is assigned to a specific set of GPS coordinates. The user can view the content when the user's GPS coordinates match. Other implementations improve tracking by incorporating a Visual Positioning System (VPS). A VPS can compare the image captured by an AR device with a library of captured images or 3D geometry of a location. Thus, the system can pinpoint a user's location with greater precision.

Some tools can also provide or even create 3D geometry of a location. This is akin to spatial mapping but for outdoor locations. This offers all the advantages of having a 3D model of the scene (see Section IX-B: Spatial Mapping). There can also be tools to simulate locations in software, complete with the 3D geometry of the scene, to simplify development. This can prove useful when creating international experiences, negating the need to travel to each location.

Fig. 13 showcases location tracking. Once the system can correctly identify its location in the real world, it can display content positioned in the real world. Here, an AR cube has been placed in front of the Eiffel Tower.

### H. PERSISTENT ANCHORS

AR content, on its own, is not persistent. Virtual objects placed will not be present for the next session. The system will have to detect, track, and place content into the scene from scratch for each session. This issue can be resolved by using Persistent Anchors. Anchors created by the user are saved with their transform and other associated data. When a new session is loaded, the system tries to place these anchors (along with their content) back into the scene. The functionality can be further improved by hosting the anchor on a network or the cloud. Persistent anchors can then be distributed to multiple devices and users, which opens the possibility for shared AR sessions.

**FIGURE 14.** Two users viewing the same AR content (cube with base) using a shared session.

### I. SHARED SESSIONS

For multiple users to share an AR session, their devices need to track the same points in space. Therefore, anchors are used as the foundation for shared sessions. By sharing the same anchor data, multiple devices can view AR content at the same point in space. Fig. 14 shows a visual representation of two users viewing the same content. Devices can also share and combine their scene maps and similar data to improve tracking. In the case of real-time shared sessions, persistent anchors are not necessary. Devices only require a means to share data. There is no requirement to save the data. Depending on the implementation, users may need to assist the system in setting it up, and thus, users need to be guided through the procedure [138], [139].

### J. LIGHT ESTIMATION

In computer graphics, rendered images that mimic the look of the real world are called photorealistic. Achieving

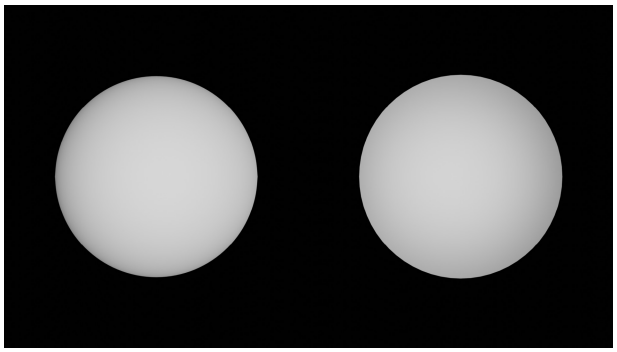

**FIGURE 15.** Two shaded circles. Despite being a sphere and a cone viewed from the top, the lighting makes it impossible to distinguish between them.

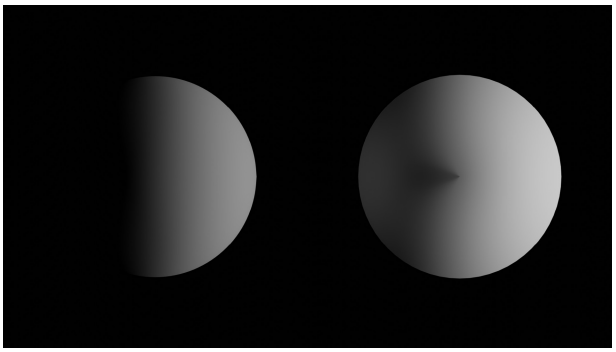

**FIGURE 16.** A shaded sphere (left) and cone (right). This is the same as Fig. 16 but with different lighting, showcasing the importance of shadows.

photorealism aligns with the interests of AR. A realistic virtual object will blend in better with the real world. The more real a virtual object appears and behaves, the more immersed a user can be in the experience. A key element of photorealism is lighting. Despite it being called lighting, it is important to note the role of shadows. Lighting is the interplay between lights and shadows. How an object is lit determines how it will appear. Fig. 15 shows two flat circles. They are, in fact, a sphere and a cone lit and viewed from the top. This can be seen in Fig. 16, with the shadows revealing the topology of the shapes.

Typically, with computer graphics, a developer has full control over the lights placed in a virtual scene and, thus, the final look. The light's size, position, and intensity can easily be adjusted as desired. However, this is a challenge for AR. The virtual object's lighting must match the lighting in the real world to blend in. The system requires some way to understand the lighting conditions of the environment. This functionality is provided by light estimation. This feature can estimate the light in the scene. It then uses this information to create a hypothetical light source that best represents the scene's lighting. This can be considered as the combination/result of all the real lights in the scene. The estimated features of this light source can then be applied to a virtual light source, which then applies the lighting to the virtual object.

Fig. 17 shows AR content with light estimation applied. Compared to Fig. 6, the lighting on the content is more consistent with the lighting in the environment. This is reflected in the shadows. The real-world lighting in Fig. 17 is coming from the left, with shadows going to the right; notice the line between the book and the metal bar to the left of the image. The shadow on the base of the AR cube has shadows with the correct orientation.

### K. ENVIRONMENT PROBE

An object reflects light from a light source as well as the reflected light of the environment. Even if present as a subtle effect, it is an important aspect of photorealistic lighting.

**FIGURE 17.** AR content with light estimation applied. The direction of lighting on the AR content is consistent with the environment.

However, reflections present a challenge for AR. Reflective surfaces can show surroundings out of the viewer's sight. Mirrors are an obvious example. In AR, this means that the system needs to display images of an area it cannot see. The solution is to use environment probes. These probes can be placed in the scene in a way similar to anchors. They are used to gather 360° views of the environment. This is then converted into environment textures such as cube maps. This texture/image information can then be used to display surroundings as reflections.

Environment probes can also be utilized to obtain lighting information since they capture a complete view of the environment. One such method is HDR cube map lighting. This can provide more realistic lighting where multiple light sources are accounted for rather than a singular estimated light source with light estimation. However, this leads to quite a bit of confusion between light estimation and environment probes. Some tools view and provide them as distinct tools. Others combine them and may even offer them under the name of light estimation or environment probes alone.

Fig. 18 showcases the same scene in Fig. 17 with Environment Probes applied on top. The AR content has been modified to be more reflective to showcase the effect better. Looking closely, one can find the Newton's cradle from Fig. 6 reflected in the AR cube (left face).

**FIGURE 18.** AR content with environment probes applied. The AR content now reflects the content in the environment.

**FIGURE 19.** AR content with occlusion applied. It is now occluded by real-world objects in front (the book and the hand), giving a proper sense of depth.

### L. OCCLUSION

Occlusion is an important feature of AR. It is an important cue for a viewer to understand where objects are placed relative to each other in the world. Occlusion between virtual objects is a problem solved by the render engine. Occlusion between real and virtual objects is more complicated. Virtual objects occluding real objects can be considered a solved problem. As virtual images are overlaid on real images, occlusion is achieved naturally. The only challenge here lay with OST devices, as virtual objects can appear translucent. Occluding virtual objects with physical objects is where the challenge lies. The system needs to identify which physical objects will occlude a virtual object. It then needs to mimic occlusion by hiding the correct parts of the virtual object. All further references to occlusion and solutions for occlusion refer to this specific challenge.

The simplest solution is to utilize an occlusion material with some clever assumptions. An occlusion material has two properties that must be applied to a virtual object. One, it makes the virtual object to which it is applied invisible. Two, despite being invisible, it must make the virtual objects with the material applied occlude other virtual objects behind it. The exact means to create such a material will vary from one rendering engine to another. The next step is to make some assumptions. Consider a scenario where a virtual object is to be displayed on an image printed on a sheet of paper. The goal is to ensure that the paper occludes the virtual object no matter which angle it is viewed from. This can be achieved by creating a virtual plane matching the size of the paper and then applying the occlusion material to it. This occlusion method relies solely on the rendering engine and is useful when the AR system alone cannot provide much information about the world.

The above solution is quite limiting as knowing or guessing information about the scene is necessary beforehand. The more adaptable solution is utilizing software features to understand the scene. Plane tracking, depth sensing, and spatial mapping are all features that can be used for occlusion. The occlusion material can be applied to the tracked plane or 3D model of the scene to achieve occlusion. Alternatively, the depth information can be processed to render parts of a virtual object selectively. Some tools also provide support for human/people occlusion. This is specifically to achieve the effect where people in the scene occlude virtual objects.

Fig. 19 showcases the application of occlusion. The AR content gets obscured by real-world objects (book at the bottom) and by a human. This lets us know that the content is placed behind the hand and the book in the real world. Without occlusion, the object would not be obscured at all. This would make it seem like the object was placed in front of the hand and the book.

### M. RECORDING AND PLAYBACK

Recording and Playback is a feature for creating AR experiences that can be viewed anytime, anywhere. The camera video stream, IMU data, and any other custom metadata are recorded and saved as a file. This file can then be played back to replicate the session. This can simplify development significantly. Developers can record a scene once and test it anywhere, reducing the need to be physically there. The same video can also be used on different devices, reducing iteration time. End users can also use this feature and have experiences on videos recorded by the API [140]. However, the second characteristic of AR is that it should be interactive in real-time. Since such experiences use a prerecorded video feed, it violates the second characteristic and cannot be considered as AR.

## X. AR STANDARDS

This section covers two standards developed for AR software. A wide variety of AR hardware exists, providing consumers with choice and competition. However, AR applications and engines would be forced to use unique and proprietary hardware APIs without a cross-platform standard. This means an application would require custom code to be deployed to each platform. This problem, also present in the VR ecosystem, is collectively termed ''XR Fragmentation'' and is the motivation behind standardization. By providing a common platform, standardization can help resolve fragmentation.

### A. OPENXR

OpenXR is an open-source, royalty-free standard created by the Khronos Group. OpenXR aims to enable developers to create applications that can run on any AR or VR system without having to rewrite or recompile their code. The OpenXR API provides high-performance, cross-platform access to XR platforms and devices [141], [142].

The standard defines a core set of features and extensions that XR devices and applications can support. Khronos provides a conformance test suite [143] and maintains a list of devices conformant with OpenXR [89]. The API is a layered architecture that acts as a mediator between the application and the device [144]. It handles the core of an XR session, including tracking, inputs, and display parameters. For the rendering and composition of images, it is to be used in conjunction with a 3D API like Vulkan [142].

OpenXR provides a lot of benefits to the XR ecosystem. Developers can easily ship their applications to multiple platforms, providing greater market reach. Vendors for XR devices/platforms can bring more applications onto their platforms for a larger library. End-users can run their apps on any system, removing vendor lock-in. This also reduces market confusion and increases consumer confidence, which helps to increase the adoption of XR technologies [142].

Despite OpenXR's infancy, the benefits are starting to show. Consider the Cross-Vendor Advanced UI extensions. This allows developers to use hand or eye-tracking UI interaction solutions from a different vendor. Commonality with UI interactions will enable users to pick up an XR platform or switch between them easily. Yet another example would be WebXR. Using OpenXR as the backend, Google Chromium can run WebXR applications on any device that supports OpenXR (see Section X-B: WebXR) [142]. Finally, development tools like Mixed Reality Toolkit 3 can be utilized on multiple platforms and devices despite being made by Microsoft [145].

OpenXR is not without its issues. Not all devices and applications are conformant to the standard. Some devices may have proprietary features or limitations that OpenXR does not cover. Some applications may use non-standard APIs or libraries incompatible with OpenXR. Additionally, the standard also requires the cooperation of vendors to truly achieve its goal. One notable name missing from the list of supporting companies is Apple [141]. That being said, OpenXR is an evolving standard. It is still in version 1.0; it remains to be seen what the future will hold.

### B. WebXR

AR experiences, typically, are distributed as applications through an app store. There are, however, benefits to distributing an AR experience through the web. Users do not need to install an app. Users can scan a QR code for a link and immediately be taken to a page with the AR session on their phones. Developers are also able to bypass restrictions from app stores. The review process by app stores can be skipped entirely and developers do not need to wait or worry about their application being accepted. The application can be deployed immediately, along with any updates. They can also avoid any fees or royalties for publishing in a specific store. The nature of the web also provides additional benefits. The web is inherently cross-platform, as web pages can deliver content across a spectrum of devices. Web apps have become more commonplace and take advantage of other web apps, services, and APIs. Web AR can do the same and integrate them into the AR application. However, Web AR has challenges preventing wider adoption. Standardized access to device sensors needs to be provided. Otherwise, it leads back to the XR fragmentation problem. Furthermore, web applications are not as performant or responsive as apps running natively on a device.

WebXR is a standard defined and maintained by the World Wide Web Consortium (W3C) to address these challenges [146]. The W3C was founded in 1994 to ensure the growth of the web [147]. The WebXR Device API provides access to XR devices' input and output capabilities. Like OpenXR, WebXR relies on graphics APIs like WebGL & WebGL2 [146]. While there are a lot of similarities, WebXR is not a 1:1 web version of OpenXR. WebXR and OpenXR are separate projects maintained by different groups [148]. WebXR also relies on OpenXR as the backend where possible [142].

WebXR provides access to many AR features. It supports most basic features: **spatial tracking (positional tracking)** [149], **plane tracking** [150], **image tracking** [151], **hit tests**, and **anchors** [146] with image tracking currently still in development [151]. WebXR also supports more advanced features like **depth sensing** and **light estimation**. It also supports multiple means of input, including hand tracking. As a web-based platform, support for **Document Object Model (DOM) overlays** is also provided as a feature (see Section XIII: Web AR Software) [146]. WebXR is cross-platform and supports smartphones, desktops, and HMDs. However, support for specific features will depend on the browser, device, and platform [146]. WebXR is meant to be used in conjunction with other tools and libraries for building the website [146]. For examples, see the section: Web AR Software.

## XI. AR FRAMEWORKS, PLATFORMS, AND SDKs

This section will cover software frameworks, platforms, and SDKs for AR development. Most, if not all, are focused on enabling an AR experience. As such, it may be preferable/required to use them in conjunction with other tools (such as game engines or DCC tools). Table 3 shows a matrix of all the AR features discussed before vs the software listed in this section. Some features are marked with an asterisk. This means the software can provide the functionality, but it is not exactly as defined in our paper or has certain limitations. We have provided explanations as to why, where possible, on a best-effort basis.

**TABLE 3.** Matrix of AR features vs. AR frameworks, platforms, and SDKs.

| Features | ARKit | ARCore | Vuforia | MRTK | Wikitude | Snapdragon Spaces | AR-media | ARToolKit |
|---|---|---|---|---|---|---|---|---|
| Positional Tracking | Y | Y | Y | | Y | Y | Y* | |
| Feature Points and Point Clouds | Y | Y | | | | Y | | |
| Plane Tracking | Y | Y | Y | | Y | Y | | |
| Hit-testing | Y | Y | | | | Y | | |
| Image Tracking | Y | Y | Y | | Y | Y | Y | Y |
| Anchors | Y | Y | Y | | Y | Y | Y | |
| Depth Sensing | Y | Y | | | | | | |
| Spatial Mapping | Y | | Y* | Y* | | Y | | |
| Scene Semantics | Y | Y | | | | | | |
| Object Tracking | Y | | Y | | Y | | | |
| Cloud Recognition | | | Y | | Y | | | |
| Face Tracking | Y | Y | | | | | | |
| Location Tracking | Y | Y | | | Y* | | Y | |
| Persistent Anchors | | Y | | | | | | |
| Shared Sessions | Y | Y | | | | | | |
| Light Estimation | Y | Y | | | Y | | | |
| Environment Probe | Y | Y | | | Y | | | |
| Occlusion | Y | Y | | | Y | | | |
| Recording and Playback | | Y | Y | | | | | |

### A. ARKit

ARKit 6 is Apple's proprietary AR framework for Apple devices and boasts an impressive feature set. The **Depth API (Depth Sensing)** takes advantage of the LiDAR Scanner on select Apple devices, opening up new features. **Instant AR** allows for instant placement of objects in the real world. The LiDAR scanner enables quick plane detection [152]. This also powers **Scene Geometry (Spatial Mapping, Scene Semantics)**, providing a 3D map of the user space with labels, e.g., floor, walls, and windows [152], [153]. This can be used for better occlusion and virtual object physics (see Section IV-A6: SLAM under Enabling Technologies: Registration). To add more photorealism, ARKit offers **Light Estimation** [154] and **Environment Texturing (Environment Probe)** [155].

ARKit has excellent features for integrating human interaction. **People Occlusion (Occlusion)** provides realistic occlusion with people moving around in the real world, improving immersion [152], [156]. Multiple types of **Motion Capture** are supported, including body [157], hand [158], and **face tracking** [159]. Hand tracking is currently limited to VisionOS (Apple Vision Pro).

In addition to supporting **all basic features** [160], [161], ARKit also supports **Object Tracking** (and **Scanning**) [162], and **GeoTracking (Location Tracking)**. Location accuracy is improved through localization imagery. ARKit downloads imagery of the physical environment in the user's area based on GPS coordinates. This imagery is then compared with the current camera image for a more precise geographic position. Localization imagery is only supported in a limited number of areas and cities [163].

Other features include **4K video capture** [152], [164], **simultaneous front and back camera** [165], **Shared AR** (multi-user sessions) [160] and **App Clip Codes** [166]. App Clip Codes are similar to QR codes and give users immediate access to critical or context-specific parts of an AR app. It also makes it easy for users to download and launch the full app. Support is also provided for NFC-integrated clip codes [166].

Apple also provides additional tools and applications for AR:

- AR Quick Look: Allows users to view 3D content in AR directly through built-in apps by Apple like Safari or Mail [168].
- RealityKit: A framework built on ARKit to develop AR experiences. It supports several features found in game engines, like photo-realistic rendering, rigid-body physics, and skeletal animations [169].
- Reality Composer: An app with an intuitive interface for developing AR experiences powered by RealityKit [170].
- RoomPlan: An API for Swift that utilizes ARKit for Scene Mapping. It returns a 3D model of the room with characteristics like dimensions and types of furniture [171].
- Reality Converter: This app makes it easy to convert more common 3D file formats into USDZ, which is the format used by Apple. It also supports previewing and editing the properties of the created USDZ file [172].

### B. ARcore

ARCore is a cross-platform AR SDK by Google. It supports AR development for Android, iOS, and Web [173]. ARCore possesses a similar set of features to ARKit. However, the implementation differs as it needs to support a greater range of devices. ARCore has a **Depth API (Depth Sensing, Occlusion)** like ARKit, but it uses a depth-from-motion algorithm to create depth images. This requires the user to move their device around as the algorithm requires views from multiple angles. Depth sensors (like ToF or LiDAR) are not always a given but can be incorporated if present [137].

This is also reflected in **Instant Placement**. While objects can be placed instantly, they are placed with an estimated

pose without the need to detect surface geometry. The pose is later updated as more tracking information becomes available [174]. For more realistic and immersive scenes, ARCore provides the **Lighting Estimation API (Light Estimation, Environment Probes)** [175] and **Occlusion** through the Depth API [137]. **Electronic Image Stabilization (EIS)** can also be utilized to deliver smoother, stable experiences even when the camera moves around [176].

ARCore supports **all basic features** [131], [173], [177] and **Augmented Faces (Face Tracking)** [178]. It also expands on Anchors with **Persistent Cloud Anchors (Persistent Anchors, Shared Session)** using the ARCore Cloud Anchor API or ARCore Cloud Anchor service. The local anchor data is uploaded to the cloud with a unique ID for the anchor. This ID is then distributed to other users and then resolved to get the anchor on their devices. This allows for persistent and shared sessions [179]. However, Google hosts the cloud service, which may be discontinued at their discretion [180].

ARCore supports location tracking through the **Geospatial API (Location Tracking)**. The device sensor and GPS data are combined with Google's VPS to match the environment for the precise location. The VPS is built on top of Street View images from Google Maps, providing wide coverage [181]. This can be used with the **Streetscape Geometry API**. This provides the 3D geometry of terrain, buildings, or other structures in a supported outdoor area [182]. Google provides the **Geospatial Creator** to support development with these APIs. This tool allows developers to preview and work with locations supported by the Geospatial API without physically being on location [183].

Incorporating AI further enhances the AR experience. The camera feed can be fed through ML Kit and Google Cloud Vision API to identify real-world objects [184]. The **Scene Semantics API** can be used to understand a user's surroundings better when outdoors. It can label and distinguish objects outdoors, such as sky, building, terrain, object, or person [185].

The **Recording and Playback API** creates AR experiences that can be viewed anytime, anywhere. The camera video stream, IMU data, and any other custom metadata can be stored in an MP4 file with the Recording API. Playing the file via the Playback API treats the MP4 file like a live session feed. This can simplify development significantly. Developers can record a scene once and test it anywhere, reducing the need to be physically there. The same video can also be used on different devices, reducing iteration time. End users can also use this feature and have experiences on videos recorded by the API [140]. However, these experiences violate the second characteristic of AR. Finally, Google also provides **Scene Viewer**, which allows users to view 3D content in AR directly through a website or an app.

### C. VUFORIA

Vuforia is PTC's cross-platform, enterprise AR solution [186]. Vuforia Engine is their primary tool for AR development. It has a free basic plan where apps are free to be published. The premium plan unlocks full use of model and area targets (see below) and removes creation limits and watermarks in the published app when using these premium features [187].

Not all basic features are supported, in addition to being present under different names [188]:

- **Ground Plane (Plane Tracking)**
- **Image Targets (Image Tracking)**
- **Device Tracking (Positional Tracking, Anchors)**

The engine expands on image tracking with additional features. **Marker tracking** is provided through **VuMarks** and **Barcode Scanner** [188]. VuMarks, in essence, are QR codes that can store data and be tracked in AR [189]. But they differ from QR codes in their ability to have custom appearances [190]. The Barcode Scanner has support for standard barcodes and QR codes [191]. Images can also be tracked on special surfaces. **Cylinder targets** allow tracking images on cylindrical objects like bottles. **Multi Targets** allow tracking more than one image arranged in regular geometric shapes like boxes [188].

**Model Targets (Object Tracking)** and **Area Targets (Spatial Mapping)** are the more advanced tracking features [188]. Area targets can take a 3D scan of an area and create an AR experience for the entire area. This can be used to create an AR experience spanning a large space, like a factory floor. In case 3D scanners are unavailable, Vuforia also supports 3D scanning through LiDAR-enabled iOS devices [192]. Another interesting feature is **External Camera** support. The engine can use any external camera to drive the AR session. It does not matter if it is connected physically to a device or has the images streamed through a network [193].

Some features are aimed at aiding development. The **Cloud Recognition Service** improves on image tracking capabilities. By leveraging the cloud, an application recognizes and tracks image targets from a database of millions. It can also simplify updating and maintaining an image database [194]. The engine also supports **Recording and Playback** [188].

Vuforia can support a large range of devices through **Vuforia Fusion**. It is Vuforia's proprietary solution to AR fragmentation [195], and essentially acts like OpenXR. Vuforia maintains a list of recommended devices, and includes the following HMDs [196]:

- HoloLens 2
- Magic Leap 2
- Vuzix M400
- RealWear HMT-1 and HMT-1Z1

As for platforms, Vuforia supports Android, iOS, Windows (through Universal Windows Platform), and Magic Leap OS. The Vuforia Engine can also be integrated for use with the game engine "Unity" [197].

In addition to developer tools for the engine, PTC also provides other applications based on Vuforia for more specialized needs:

- Vuforia Expert Capture: A Software-as-a-Service (SaaS) AR solution. Experts can capture their work routine with an AR device. This captured expert knowledge can then be played back as instructions in AR to other workers [198].
- Vuforia Studio: An easy-to-use, low/no-code version of Vuforia Engine. It lacks support for a few features compared to the engine [199].
- Vuforia Chalk: An application for providing AR-based remote assistance [200].

### D. MIXED REALITY TOOLKIT (MRTK)

Mixed Reality Toolkit (MRTK) is a cross-platform development kit for MR applications. It is an open-source project by Microsoft. The goal is to provide cross-platform input systems and basic building blocks for the User Experience (UX) in MR [145]. MRTK does not offer many of the AR features discussed before. It focuses more on improving the AR experience than enabling an AR session. There are three versions of MRTK available (in order of increasing feature set):

- MRTK for Unreal [201]
- MRTK2 - Unity [202]
- MRTK3 - Unity [145]

MRTK for Unreal consists of two parts: **UX Tools** and **Graphics Tools** [201]. UX Tools consist of code and example assets for common features in UX development for MR. The included UX building blocks enable developers to add common MR interactions easily. Many of them support hand interaction [203].

Graphics Tools aim to improve the visual fidelity of MR applications. It provides useful visual effects such as clipping primitives and spatial perception. Clipping Primitives allow one to dynamically slice away into a 3D model, providing a better look at what is inside. Spatial Perception is a set of techniques to make more compelling materials that can be applied to the 3D scene map [204].

There is a significant increase in features with MRTK2 for Unity. It expands on UX building blocks and supports eye-tracking interactions. **Spatial Awareness(Spatial Mapping)** is now built into the toolkit [202]. This specific functionality is likely limited to the HoloLens 2. The built-in **input system** supports multiple means of input for various platforms [205]. **Extension services** can extend the functionality of MRTK. They can be created or be provided by MRTK or other parties [206].

MRTK also supports features to aid device-agnostic MR app development. The **camera system** is used to configure the application's camera, i.e., the display view. This system allows an app to support both OST and VST displays [207]. **Profiles** are used to configure MRTK and initialize the subsystems and features according to the device. This negates the need to configure an app for each device [208].

MRTK 3 for Unity is the latest iteration. It has multiple improvements and additions over MRTK2, but not all features have 1-1 correspondence or have been ported over. As such, porting applications from MRTK 2 to 3 is not recommended. It adds **accessibility features** and improves the architecture and performance of the toolkit [145].

MRTK does not support Recording and Playback, but all versions of MRTK support **Input Simulation**. This allows developers to use a keyboard and a mouse to simulate AR inputs, such as HMD pose or hand tracking [209], [210], [211].

MRTK's feature support for other platforms depends on the version used. MRTK-Unreal should support any device with OpenXR for UXTools. But it has only been tested on HoloLens 2 and Windows Mixed Reality VR devices [203]. Graphics tools only support HoloLens 2, Windows, and Android [204]. MRTK2 supports OpenXR, Windows XR, and Oculus in addition to ARKit and ARCore via Unity's ARFoundation [202]. MRTK3 is built to be more OpenXR-focused. However, support for OpenXR devices outside of HoloLens 2 is experimental [145].

### E. WIKITUDE

Wikitude is a cross-platform AR SDK owned by Qualcomm [212]. The SDK has two editions: Professional and Expert. The Professional Edition supports Unity, JavaScript, and Native development. The Expert Edition offers more features by leveraging Unity while limiting development to the engine [213].

Both editions support **single and multiple image tracking**, as well as **cloud recognition**. They also support **single object/scene tracking** [214]. Scene tracking is used for recognizing structures larger than table-sized objects, such as rooms or faces of buildings [215]. Features exclusive to the professional SDK include **Geo AR (Location Tracking)** and **Instant Tracking (Anchors)**. Geo AR can support single or multiple location-based markers/Points of Interest (POI). It is only available with the JavaScript API [214].

Instant Tracking in Wikitude is different from Instant AR in ARCore or ARKit. Instead, it enables tracking without the need for targets such as images and objects. This is powered by Seamless AR Tracking (SMART) API, which integrates ARKit, ARCore, and Wikitude's SLAM into one. **Plane Detection (Plane Tracking)** is available through Instant Tracking [216]. Instant Tracking also enables Anchors through **Extended Tracking** [214]. However, extended tracking has been marked as deprecated since version 9.12.0 [217].

The Expert Edition supports multiple tracking for object/scene tracking. It also allows mixing multiple tracking types. Image tracking is further expanded with **single and multiple cylinder tracking**. It uses AR Bridge to integrate ARCore and ARKit. AR Bridge also expands the feature set by leveraging Unity's ARFoundation. The expert edition also offers Universal Render Pipeline (URP) support [214].

Unity's URP is aimed at creating optimized graphics for a wide range of platforms [218]. The expert SDK further supports **Light Estimation**, **Environment Probes**,

**Human Depth Occlusion**, and **Camera Grain** through ARFoundation. Camera grain can add noise in the real captured image for the virtual objects. This improves the blending between real and virtual images [219].

Wikitude also provides Wikitude Studio, a tool for easily creating AR experiences without needing deep technical or programming skills [220]. It is a free tool and does not need a license key. However, there are extra benefits to buying the SDK. You can use Studio to create image targets (.wtc files) and object targets (.wto files), which can be published to the Cloud. One can also create Studio Editor projects that can be exported and integrated within an app or uploaded on a server [221]. A separate encoder tool allows 3D model files (.fbx) to be converted into .wt3 files used by Wikitude [222].

Wikitude has extensions for development for Xamarin, Flutter, and Cordova. Wikitude can be deployed to Android, iOS, and Windows devices. For HMDs, Wikitude supports devices by Epson (Moverio) and Vuzix, as well as the HoloLens [214]. Wikitude is a paid SDK with three pricing plans: standard, cloud, and enterprise. The standard and cloud plans have an annual fee per app, with custom fees for enterprises. The standard plan is free for Snapdragon Spaces developers [223]. It is unknown if the Expert Edition is available with all plans. While URP is listed on all plans [223], it is excluded from standard and cloud plans in the feature list [214].

### F. SNAPDRAGON SPACES

Snapdragon Spaces is an XR SDK by Qualcomm for XR HMDs based on Android [224]. The SDK supports **all basic features** [225], [226]. For advanced features, the SDK supports **Spatial Meshing (Spatial Mapping)** [226]. It also supports additional means of input: **gaze control**, **hand tracking**, **handheld XR controllers**, and **companion controllers**. The **Companion/Host controllers** allow a smartphone to be used as a 3DOF controller [227]. It is currently meant to be used with HMDs tethered to smartphones.

The SDK is meant to be used with either Unity or Unreal Engine. The list of supported HMDs is currently limited to two: Lenovo Think Reality A3 and Lenovo ThinkReality VRX [224]. The SDK does utilize OpenXR [228], which could help support more devices in the future.

### G. AR-MEDIA

AR-media by Inglobe Technologies is a zero-code MR platform. Supported features are limited to **image tracking, geolocation (Location Tracking), and Spatial Tracking (Positional Tracking, Anchors)**. Spatial tracking requires ARKit or ARCore to function [229]. AR-media provides three tools for creating and delivering AR experiences [230]:

- AR-media Studio
- AR-media Plugins
- AR-media Player

AR-media Studio is a web-based tool for creating, managing, and distributing projects. Content can be imported directly into AR-media Studio. It can also be imported through AR-media Plugins for 3rd party DCC tools [230]. Plugins are available for SketchUp and 3ds Max [229].

The projects are then deployed to the AR-media Player, an app for Android and iOS [230]. AR-media offers multiple pricing plans, with a free plan for individuals. The project content is hosted online, and higher-tier plans support more projects and content. They also offer more features. The plans for firms and enterprises support publishing white label apps (with additional charges) [231]. This will allow developers to publish their AR project as an app under their brand instead of the AR-media Player.

### H. ARToolKit

ARToolKit began as a software library for AR applications [232]. It possesses a limited set of features and at best (with artoolkitX) offers only **Image and Marker Tracking**. **Image tracking** is supported through NFT. For **Marker tracking**, it supports barcode and pictorial/pattern markers. It also supports **Multi-Marker tracking** [233]. However, the markers must be in a fixed position relative to each other. The markers cannot be moved individually in a session [234].

ARToolKit is still worth mentioning despite its limitations for three reasons:

- historical significance
- variety in supported OS, software, and languages
- open source code

ARToolKit was first developed in 1999 by Dr Hirokazu Kato at HIT Lab [235]. ARToolworks was founded in 2001 to develop ARToolKit further and provide commercial licenses and solutions [236]. ARToolworks was sold in 2015, which led to the creation of artoolkitX. This was to ensure that ARToolKit would continue to be developed and supported into the future [237].

Its long history has led to ARToolkit supporting a variety of OS, software, and programming languages. artoolkitX supports iOS, Android, macOS, Windows, and Linux [238]. artoolkitX also provides a version for use with Unity [239]. Currently, artoolkitX also maintains JSARToolKit5. It is based on JavaScript and can enable web-based AR experiences [240].

More variants of ARToolkit can be obtained from ARToolworks. These include versions for Java and niche ports like those for Silverlight or Adobe Flash [241]. Most software under ARToolworks has a dual license model: one for open source and one for commercial. The open-source license requires developers to make their applications open-source as well. The commercial license does not have this requirement [242].

The different variants of ARToolkit and its open-source nature make it an excellent starting point for building a custom AR library from the ground up. AR.js, for instance, is built on top of JSARToolKit5 (see Section XIII-B: AR.js).

Should the need ever arise, the variants will also allow development for niche or perhaps obsolete platforms.

### I. OTHER POSSIBILITIES

There are a few other possibilities we could not discuss in detail that can be considered:

- Niantic Lightship [243]
- Meta Spark Studio [244]
- TikTok Effect House [245]

## XII. GAME ENGINES FOR AR

Game engines are tools used to simplify the task of game development. Games can be built with little technical expertise, with the engines handling the more technical aspects, such as rendering or physics simulation. A developer's task can be simplified to producing assets (like 3D models or images) and assembling them with appropriate systems for logic and behavior. Many of the features they provide for video games, particularly in 3D, have applications in other fields. Game engines have since expanded to support such fields, including AR. Some have even adopted the term "real-time 3D" engine, as it is no longer solely for game development.

There are many reasons to utilize a game engine for developing AR applications. Many features and tools that one might need for complex 3D experiences have already been implemented in game engines. Combining an AR framework with an appropriate game engine brings more features to the table to build more complex experiences. AI state machines can be used to build reactive virtual characters. Animation state machines and blending can be used to build reactive animations. The built-in physics engine can enable physics for virtual objects. Additionally, game engines often leverage the Entity-Component System (ECS) software architecture. Games are usually complex systems, which complicates the task of development. By using the ECS architecture, a game can be broken down into smaller systems of entities and components. The use of entities and components provides modularity and reusability. It is a proven architecture that forms the foundation of most, if not all, engines.

Before we begin, there are a few things to note with this discussion. No AR software features will be listed in this section. Game engines rely on other AR frameworks to provide AR functionality. Therefore, any supported features will depend on the framework used with the engine. Similarly, supported hardware is left out of discussion as it depends on the framework. Finally, game engines are often customized to the needs of the developer. There are official and 3rd party tools and extensions that will be available. As such, these engines may support other programming languages or frameworks and are not limited to what is mentioned.

### A. UNITY

Unity is a real-time 3D development engine by Unity Technologies [246]. It uses C# as the native development language [247]. It offers a lot of features for AR development [248]:

- AR Foundation
- Unity MARS
- XR Interaction Toolkit
- Unity as a Library

AR Foundation is Unity's solution to XR Fragmentation. The framework combines core features from ARKit, ARCore, Hololens, and Magic Leap. This, combined with Unity's features, provides a unified workflow for AR development. Not all features are available on every platform. However, Unity has the means to easily integrate an unsupported feature should it be enabled in the future. AR Foundation can also be integrated with Unity MARS [249].

MARS is a paid extension that simplifies AR development and reduces development time. MARS supports cross-platform development for iOS, Android, and the HoloLens [250]. It aims to address the challenges of developing AR apps for dynamic physical environments. MARS provides plain-language authoring where app behavior can be described with plain language, reducing the amount of code required. The proxy-based workflow provides proxies to represent real-world objects. This makes development more agnostic to the specifics of the physical environment [250].

MARS can simulate environments in Editor, reducing iteration time. Templates with customizable components are provided for faster prototyping. An AR companion app is provided for Android and iOS. It can be used to capture environments and scan objects [250]. Object scanning is limited to the Unity Editor for Mac. It can also do some minor editing of the project, with changes reflected in the editor [251].

XR Interaction Toolkit is Unity's solution for cross-platform interaction systems for XR. Commonly used interactions are provided as premade components, similar to MRTK [252]. MRTK 3 was built on top of Unity's XR Management system and XR Interaction Toolkit [145].

"Unity as a Library" allows developers to insert features of Unity directly into a native app. This can be used to embed AR experiences within existing mobile apps without the need to rebuild them from scratch. This functionality can also be expanded to applications for Windows [253].

Unity's greatest advantage (over other engines) is its comparatively well-developed AR integration, excellent documentation, and massive community. Unity supports many AR frameworks and tools, including OpenXR, ARKit, ARCore, Vuforia, WebXR, and MRTK. For HMDs, it has official support for HoloLens 2 and Magic Leap 2.

Unity operates on a subscription model but offers free plans. Paid plans mostly provide increasing support and guidance for Unity as far as AR is concerned [254]. MARS is included in the Unity Pro, Enterprise, and Industry plans. Other plans require an annual subscription at $50/month. All plans are royalty-free and include access to the Unity Asset

Store (asset marketplace) and Unity Learn (training materials by Unity) [254].

### B. UNREAL ENGINE

Unreal Engine 5 (UE5) is a real-time 3D tool by Epic Games [255]. UE5 relies on C++ and Blueprints for programming and scripting. Blueprints are a visual scripting system relying on classes defined by C++ [256]. UE5 supports AR for Android, iOS, and the HoloLens 2 out of the box [257].

UE5 provides a unified framework for handheld AR experiences (ARCore and ARKit) [258]. Development for the HoloLens 2 is supported through OpenXR [259]. Support for Magic Leap 2 could be expected in the future, as Magic Leap 1 was supported in UE4 [260]. UE5 supports shared AR sessions through ARPins (persistent anchors) [261].

Unreal also provides tools based on AR frameworks. These tools could prove useful for AR development. RealityScan is an app for Android and iOS that can be used to scan real-world objects to produce 3D models. It is technically free to use but relies on Sketchfab to export models. Sketchfab operates on separate pricing plans from Unreal, limiting uploads [262]. The Live Link Face app, powered by ARKit, can easily capture facial animations. The usage of ARKit limits the app to iOS [263]. This app can be combined with MetaHuman, a tool for creating and animating realistic human models [264]

Unreal Engine is open-source and free to use. It charges a royalty of 5% after a project earns over $1 million. Alternatively, other licensing solutions are available providing direct support services from Epic Games [255].

### C. GODOT

Godot 4.0 is a completely free, open-source game engine maintained by the Godot Foundation [265]. Godot offers support for four programming languages: GDScript, C#, C++, and C. GDScript is a custom language built for Godot, with syntax similar to Python. C and C++ are supported through GDExtension [266].

OpenXR is now built into Godot 4.0 and does not require a plugin. It supports Magic Leap 2 and Lynx R1 for AR HMDs. Godot 4 also provides Godot XR Tools, a toolkit providing popular XR mechanics [267]. Support for ARKit and ARCore is currently not listed for Godot 4 but can be expected as Godot 3 supported them both [268]. Godot 4 was released in 2023 [267] and it will be a while before support for additional frameworks is added. In the meantime, Godot 4.0 supports WebXR [267], which can be used as an alternative for AR on handheld platforms.

Godot still has a long way to go to match the developed status of other engines. But it is open-source and free to use. It is also incredibly lightweight for a game engine. The base editor is distributed as a standalone application [269] with a file size of less than 150 MB (at the time of writing). In addition to the three major desktop OS (Windows, Mac, and Linux), Godot also has editors for Android and the web [269]. All this means that the Godot Editor can run quite smoothly on lower-end devices.

## XIII. WEB AR SOFTWARE

This section covers frameworks, libraries, and other tools specifically meant for web-based AR development. Table 4 shows a matrix of all the AR features discussed before vs the software listed in this section. There are two things to note in Table 4. First, quite a few web AR software rely on WebXR as their foundation. Thus, despite not being in this section, WebXR has been included in the matrix. Second, we added DOM overlays to the list of features. Document Object Model (DOM) is "the data representation of the objects that comprise the structure and content of a document on the web" [270]. DOM overlays are a way to overlay DOM content on the user's visual feed. Thus, DOM content can be repurposed for use in AR, and this feature is often used to create a UI quickly [271].

### A. MODEL VIEWER

Model viewer (stylized as < model-viewer>) is a web component to easily add 3D models for display on a web page or in AR through HTML [272]. It supports AR through 3 modes:

- WebXR for Android
- SceneViewer (see Section XI-B: ARCore) for Android
- QuickLook (see Section XI-A: ARKit) for iOS.

WebXR is the default mode for Android [273].

The supported WebXR features are limited to **hit-tests** and **DOM Overlays**. WebXR's AR features are currently limited to Chromium Browsers [272]. Additional post-processing effects can be added through the < model-viewer-effects> library addon [274]. A model editor is provided to quickly configure and test the model viewer experience [275].

### B. AR.JS

AR.js is a lightweight library for AR on the Web. It is fully based on JavaScript and offers three AR features: **Image Tracking**, **Marker Tracking**, and **Location Based (Location Tracking)** [276]. Marker tracking is further split into two: Barcode or Pattern [277]. It also offers **DOM overlays** [278].

The library can work on any browser with support for WebGL and WebRTC. It utilizes jsartoolkit5 for the tracking. For rendering and display, AR.js depends on two solutions:

- three.js (see Section XIII-F3: three.js under Web AR Software: Other Possibilities)
- A-Frame (see Section XIII-C: A-Frame)

AR.js maintains a separate version for each solution [276]. While intended for use with smartphones and tablets, AR.js can also work on laptops. It is unknown whether it can function on HMDs through browsers.

It should be noted that at the time of writing, the A-Frame version has issues with resizing. Rotating the image with

**TABLE 4.** Matrix of AR features vs. web AR software.

| Features | Web XR | Model Viewer | AR.js | A-Frame | Babylon.js | 8th wall |
|---|---|---|---|---|---|---|
| Positional Tracking | Y | | | | Y* | Y |
| Feature Points and Point Clouds | | | | | | |
| Plane Tracking | Y | | | | Y | |
| Hit-testing | Y | Y | | Y | Y | |
| Image Tracking | Y | | Y | | | Y |
| Anchors | Y | | | Y* | Y | Y |
| Depth Sensing | Y | | | | Y | |
| Spatial Mapping | | | | | | |
| Scene Semantics | | | | | | |
| Object Tracking | | | | | | |
| Cloud Recognition | | | | | | |
| Face Tracking | | | | | | Y |
| Location Tracking | | | Y | | | Y |
| Persistent Anchors | | | | | | |
| Shared Sessions | | | | | | |
| Light Estimation | Y | | | Y | Y | |
| Environment Probe | | | | | | |
| Occlusion | | | | | | |
| Recording and Playback | | | | | | |
| DOM Overlay | Y | Y | Y | Y | Y | |

respect to the camera changes the transform of the AR content with respect to the image. With markers, resizing the browser after displaying AR content will affect its transform. It will return to normal after the browser returns to its original size. This can occur when changing from portrait to landscape on a phone.

### C. A-FRAME

A-Frame is a cross-platform web framework for XR [279]. It is based on three.js with an ECS architecture [280]. A core feature and strength of A-Frame is its use of declarative HTML. An XR experience can be set up by simply importing in the A-Frame script and declaring an A-Frame scene tag in HTML [279].

By leveraging the provided entities and components, AR experiences can be quickly prototyped with a few lines of HTML [281], [282]. A-Frame supports AR.js and all its features [276]. A-Frame also supports WebXR [283], but support is limited to **hit-tests** [281], [284], **DOM overlays** [281], [283], **light-estimation** [285], and **anchors** (through hit-test).

A-Frame is not limited to just HTML. Further functionality can be added through Javascript, DOM APIs, three.js, and WebGL. The ECS architecture allows for easy integration of community-made components and entities. To further ease development, A-Frame provides a built-in visual inspector to view the workings of the scene [279].

### D. BABYLON.JS

Babylon.js is an open-source game engine for the web, bringing advanced rendering and physics simulation to the web [286]. It offers an impressive feature set (considering the web), as well as integration for other tools. These include 8th Wall, Unity, 3ds Max, Maya, and Blender [287].

Babylon.js relies on WebXR for AR. The supported WebXR features include **hit-test, DOM Overlay, anchors (by extension, positional tracking), plane detection (Plane Tracking), light estimation,** and **depth sensing** [288]. Since Babylon.js has support for the HoloLens 2 [288], it also has a port of MRTK [289].

Babylon.js also provides a suite of tools to help aid with development. The Sandbox can be used to preview 3D models or .babylon files online [290]. The Playground can be used to experiment with code. Code can be written in JavaScript or TypeScript, with the results rendered in a preview window [291]. The Node Material Editor can be used to create GLSL shaders through visual scripting, simplifying the process [292]. Finally, to simplify GUI creation, a GUI Editor is also provided [293].

### E. 8TH WALL

8th Wall is a paid platform for creating WebAR experiences [294], [295]. Supported features include **positional tracking, image tracking, anchors, face tracking, light estimation, instant surface detection,** and **Lightship VPS (Location Tracking)**. **Image tracking** can support flat, cylindrical, and conical surfaces **Lightship VPS** can provide a 3D mesh of the location for occlusion and physics [294]. **Hand tracking** has been introduced with Release 23 [296]. Support for shared AR is also available as module [297].

8th Wall provides a cloud-based editor for creating applications. The editor supports web frameworks like React and Vue.js. It also supports remote debugging, instant preview, and support for external libraries. 8th Wall also supports other 3D web frameworks, namely A-Frame, three.js, Babylon.js, and PlayCanvas [294].

8th Wall also provides additional services for deployment and distribution. It provides deployment management as well as global web hosting services for projects [294]. 8th Wall

can also generate a QR code (QR 8Code) and shortened link for easier distribution of the project [298]. 8th Wall is not limited to web browsers and can support other apps like Slack or Gmail. With Link-out support, 8th Wall can direct apps without native support for certain AR features to apps with the support [299].

8th Wall has its proprietary implementation for AR utilizing other libraries [294], [295]. However, it also leverages other AR frameworks, the extent of which is not fully known. ARCore has been listed amongst the open source licenses [300]. WebXR has been utilized for VR projects with 8th Wall [301]. While there is no mention of ARKit, it is unlikely it has been left out.

### F. OTHER POSSIBILITIES

There are a few other possibilities we could not discuss in detail that can be considered:

#### 1) NEEDLE ENGINE

Needle is a web-based runtime for 3D apps [302]. It claims theoretical support on all devices supporting WebXR. Devices tested and confirmed to be supporting AR include Android (10+), iOS (15+), HoloLens 2, and Magic Leap 2 [303].

#### 2) PLAYCANVAS

PlayCanvas is a WebGL game engine with a vast feature set like Babylon.js [304]. It supports AR applications through WebXR, with support provided for other frameworks like 8th Wall and ZapWorks. With WebXR, AR can be run on Android with Chrome. WebXR support for iOS is still a work in progress [305].

#### 3) THREE.JS

three.js is a JavaScript library for rendering. It possesses no AR capabilities of its own and relies on other libraries like WebXR for running the AR session [306]. three.js is nevertheless important as it forms the foundation for rendering with many other web AR solutions. three.js maintains a list of compatible libraries and plugins [307].

#### 4) P5.XR

p5.xr is an add-on for p5.js [308]. p5.js is a JavaScript library made to make coding more accessible with a focus on creating visual projects [309]. Currently, support is claimed to be limited to Google Chrome and devices supported by ARCore [310]. This limits support to Android.

#### 5) REACT-XR

React-XR (or react-three-xr) is a React library providing React components and hooks for react-three-fiber [311], which is a react renderer for three.js [312]. React-XR relies on WebXR to run the XR session. Its viability for AR could not be confirmed at this time.

#### 6) VERGE3D

Verge3D is a paid toolkit for creating interactive web experiences. Its primary selling point is Verge3D Puzzles, a visual scripting language that allows non-coders like artists to add logic and behavior to an experience [313]. This is very similar to the coding language Scratch. Verge3D supports AR through WebXR [314]. The Verge3D toolkit is available for Blender, 3ds Max, and Maya, all of which are DCC tools. Three licensing models are available, ranging from $290-$2990 [315]. An all-in-one package is available as the ''ultimate edition'' for $3990 [316].

#### 7) ZAPWORKS

ZapWorks is a platform for Web AR from Zappar. It provides SDKs for Unity and other web AR software in addition to tools for AR development [317]. It appears to have a proprietary system for AR [318] (like 8th Wall), but it also has support for WebXR [319]. Pricing ranges from $68-$458/month, and there are alternative licensing options [320].

#### 8) WONDERLAND ENGINE

Wonderland Engine is a platform for developing web-based graphics applications with the provided Wonderland Editor [321]. Wonderland supports AR through both 8th Wall and WebXR [322]. Wonderland Engine is free to use [323].

## XIV. GUIDELINES FOR AR DEVELOPMENT

In this section, we go over guidelines for AR development. Fig. 20 provides a visual breakdown of the topics and structure of discussion in this section. This section is the result of an informal thematic analysis. Information was extracted from seven primary sources and manually tagged in the note-taking application ''Obsidian''. The graphing functionality in Obsidian allows data to be linked and visualized through graphs (see Fig. 21. Tagging the information allowed them to be grouped under common themes and then be ''distilled'' to produce the set of guidelines in this paper. This process aimed to reduce redundant information and provide a structure to these guidelines. This has been done to the best of our ability, as the interconnected nature of the information (see Fig. 21) makes it difficult to isolate them into groups for categories. This is particularly evident in the guidelines for display (see SectionXIV-E: Display under Guidelines for AR Development).

### A. DESIGN THINKING

In this section, we go over the steps to generate and validate ideas for an AR application. We utilize the principles and steps of design thinking. They are very helpful in creating any product, and AR is no different.

#### 1) BRAINSTORMING

Before brainstorming, the first step would be to conduct research and to interview people to identify their problems in the real world that AR can address [324]. The problem,

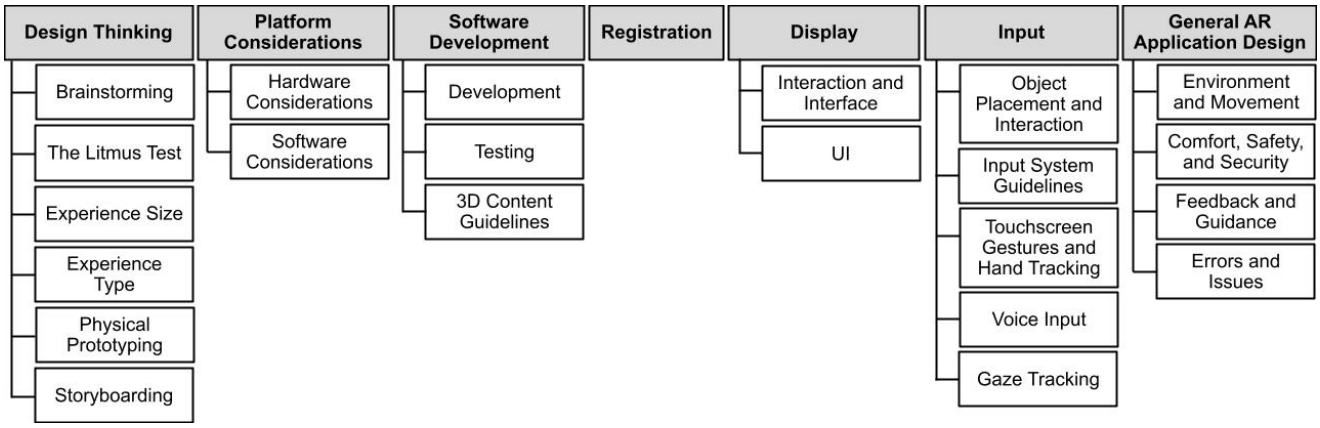


**FIGURE 20.** A visual breakdown of the section "Guidelines for AR Development". Primary topics are at the top (gray) and go from left to right.

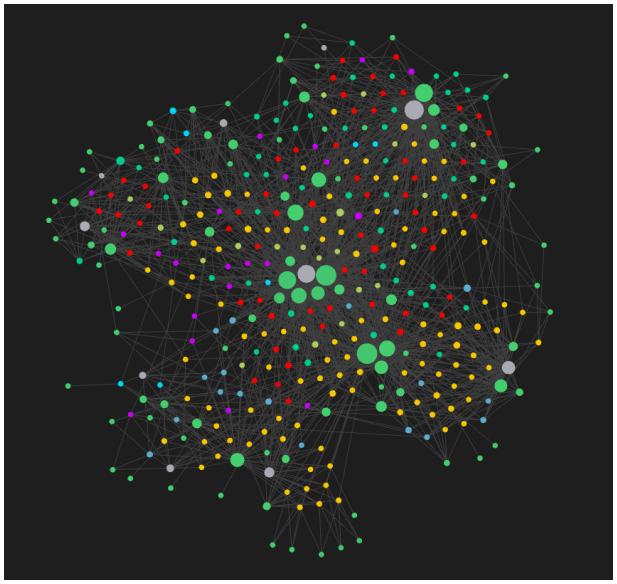

**FIGURE 21.** The graph in Obsidian with the guidelines extracted from various sources. The lines depict relationships between nodes. Green circles represent tags, that were later used to group guidelines (large gray circles). The smaller circles represent the guidelines, with colors assigned by the source of the guidelines.

preferably, should only be addressable in AR for reasons we will discuss (see Section XIV-A2: The Litmus Test). Once the problem is identified, the next step is to brainstorm possible solutions. Here are a few tips for brainstorming [325]:

- Stay on topic
- Avoid being too detailed
- Go for quantity
- Encourage wild ideas
- Be visual, use diagrams or props
- Note down all ideas
- Build on the ideas of others
- Have one conversation at a time and listen
- Defer judgment until the end

These guidelines are important because the first few ideas generated are often safe and obvious. Pushing the boundaries increases the chance of a more novel solution. After brainstorming, it is time to narrow down possible candidates. Typically, the next step in design thinking would be to proceed with physical prototyping. However, as indicated by the next few sections, we believe a few more tasks need to be addressed before physical prototyping.

### 2) THE LITMUS TEST

The novelty of a technology, actual or apparent, can sometimes be blinding. It is easy to get caught up in the hype and ultimately end up with a gimmick instead of a real solution. Gimmicks may have appeal, but they lack intrinsic value. As the facade drops, the appeal will fade, and a disillusioned user base will abandon such gimmicks. Failure to distinguish between a gimmick and a real application by a project or company could very well be their death knell if not accounted for by strategy. AR is not impervious to this effect, but we have developed a litmus test to help distinguish between application and gimmick: "A great AR application lets you do something impossible or impractical in reality."

The point of AR is that it augments reality. If an AR application performs a function that can be achieved more efficiently by other means, it has missed the true strength of AR. In a technical sense, the ability to augment reality is unique to AR. Thus, an AR application achieves something that cannot be done by other means. However, the existence of alternatives is a testament that the problem does not require AR. This can diminish the value of using AR and introduces the need to compete with alternatives. This, in turn, leads to the question of practicality. AR has plenty of issues to contend with. Many of those issues are strikes against using AR. In addition, if there are alternatives with more advantages in practice, the value of AR will be diminished further. An AR application, therefore, needs to be practical enough to overcome its weaknesses and survive its competition. AR has appeal, but value distinguishes it from application and gimmick.

This is not to say that "gimmicks" are not to be made. Value is subjective. Sometimes, the appeal alone is the value of a product or perhaps even its function. Also, an application that is largely a gimmick can still have its uses. For example, a measuring tape can easily beat an AR measurement tool. The measuring tape even has advantages, such as being able to measure around round surfaces and areas that are occluded. However, the AR application becomes valuable in situations where one does not have a tape measure at hand. After all, most people carry a phone rather than a measuring tape.

#### 3) EXPERIENCE SIZE

The size of the experience is an important factor that affects the application's planning, design, and feel. For instance, placing large AR objects in an insufficiently sized room can lead to depth collisions [138]. This will ruin the immersion of the experience. It is important to let users know the experience's size before engaging in it. Users may open the application without enough room to move around or enough large and flat surfaces typically used for object placement. Communicating the application's requirements and expectations to people upfront will help them understand how their physical environment can affect their AR experience. A responsive playspace that adapts to the environment with differing feature sets can make the experience more flexible to the environment [135], [139], [324], [326], [327]. However, experience size does not necessarily have to be constrained by the environment. Content can still be in the world out of view. Users should be given the time and ability to adapt to such an experience [328]. Audio and visual cues can be used to direct user attention to content outside their view. Alternatively, larger objects can be added to the experience by placing them a good distance away or by "breaking" reality and expanding into a virtual environment [329], [330]. Once the entire object is shown to the user, it can then be brought close to the user to reveal its true scale [328]. These "out-of-view" techniques can also have other applications (see Section XIV-G1: Environment and Movement under Best Practices for AR Applications).

#### 4) EXPERIENCE TYPE

It is important to know the experience type to settle on the experience size. Knowing the experience type will help in planning and designing the application. Experience type can be classified and defined in many ways, but experiences need not be limited to any one type. For instance, consider one classification by Microsoft. Applications are classified on how the virtual objects are blended with the environment [331], similar to the Reality-Virtuality Continuum. While this is valid as a form of classification, we believe it is too broad in definition and thus, does not provide the necessary constraints to help design an AR application. Therefore, we focus on classifications defined specifically by user interaction and the space they'll interact in.

Microsoft has another classification that suits our requirements, where experiences are classified by their "scale" (see Table 5). This classification by Microsoft was designed with the principle of an experience scale supporting the applications (and, by extension, features) of a smaller scale. The requirements of a smaller scale also apply to a larger scale. Table 6 shows how the features of a lesser scale are nested within the larger scales [136]. With an orientation-only experience, the content uses an attached frame of reference [136]. An attached frame of reference is locked/attached to a position and/or orientation, typically to some part of the user. In orientation-only scale, the content moves with the user's body, with the user free to look around with their head. An attached frame of reference can act as a backup when other frames are unavailable. For example, when the device cannot find its position with reference to the world, content can always be anchored to the device by choosing it as the origin [136].

With the seated-only scale, it uses a stationary frame of reference. A stationary frame of reference is used to create world-locked content, where content remains in the same place even as the user's position changes [136]. All scales from seated-only and beyond use the stationary frame of reference. The primary difference between them is the size of the stationary reference frame. Standing-only is similar to seated-only, but the coordinate system defines the floor/stage on which the user stands. Expanding further, room-scale defines a region of space where the user can move around, while world-scale attempts to provide the room-scale experience without bounds or limits. Microsoft may list separate frames of reference for each scale, but this is because additional requirements need to be fulfilled to provide and expand the stationary frame of reference.

It is important to note that the above classification system was originally made with HMDs in mind. The above classification can also be utilized for handhelds and other AR platforms, provided appropriate adaptations are made.

#### 5) PHYSICAL PROTOTYPING

Once we have settled on the idea and the general size and type of the experience, we can begin physical prototyping. By physical prototyping, we mean prototyping without AR hardware and software. Instead, we use props and other physical materials to simulate the AR experience. But why prototype physically? Why not directly create and test prototypes with AR applications? Because prototyping outside of AR is cheaper and quicker [332], [333]. When prototyping in code, one has to deal with errors in code, build times, and bugs, which get in the way of properly identifying problems with the application's design. With physical prototyping, we can identify the design and features we need before we commit to any level of technical development. Traditionally, application design is for a 2D context. However, 2D design philosophies start to break down as we move into 3D. While 3D is best represented through

**TABLE 5.** Mixed reality experience scales [136].

| Experience scale | Requirements | Example experience |
|---|---|---|
| Orientation-only | Headset orientation (gravity-aligned) | 360° video viewer |
| Seated-scale | Above plus headset position based on zero position | Racing game or space simulator |
| Standing-scale | Above plus stage floor origin | Action game where you duck and dodge in place |
| Room-scale | Above plus stage bounds polygon | Puzzle game where you walk around the puzzle |
| World-scale | Spatial anchors (and typically spatial mapping) | Game with enemies coming from your real walls |

**TABLE 6.** Nesting of features within larger scales [136].

| 6DOF tracking | Floor defined | 360° tracking | Bounds defined | Spatial anchors | Max experience |
|---|---|---|---|---|---|
| No | - | - | - | - | Orientation-only |
| Yes | No | - | - | - | Seated |
| Yes | Yes | No | - | - | Standing - Forward |
| Yes | Yes | Yes | No | - | Standing - 360° |
| Yes | Yes | Yes | Yes | No | Room |
| Yes | Yes | Yes | Yes | Yes | World |

3D applications, these applications can require technical expertise. Thus, prototyping and iteration can be bottlenecked by the lack of expertise across the entire team. But with physical prototyping, everyone can engage in it regardless of expertise [332]. Simple and cheap art, office, or workshop supplies, as well as certain toys, can be useful for prototyping. Consider the following list:

- Styrofoam shapes (discs, cubes, spheres, cones)
- Cardboard boxes
- Cardstock, sticky notes, plain paper
- Wooden dowels, popsicle sticks
- Disposable cups
- Tape, glue, glue guns
- Scissors, hobby knives, craft blades
- Paperclips, filing clamps, staples
- Twine, thread, yarn
- Pencils, sharpies, highlighters
- Meccano, lego, animation armatures

Once the physical prototype is ready, acting out the experience is a great way to gauge the experience before building anything. It will help get into the user's mindset and notice obvious issues or constraints with the design. The acting session can be recorded to provide guidance later in the project, providing a visual example of what works and what does not [332]. Before moving on to software development, be it digital prototyping or asset creation, the results can then be translated into a storyboard [324], [332].

#### 6) STORYBOARDING

Storyboarding is a common technique in the entertainment industry, particularly for movies and video game development. They are almost like a comic book, showcasing the actions and the experience to be built. Storyboards can be either low or high-fidelity. Low-fidelity storyboards are simple and used to convey ideas quickly, capable of being made by anyone. High-fidelity storyboards can be well-detailed and colored. It exists to showcase the final product and its aesthetics. Its primary goal is to sell the idea of the final application to important people like stakeholders [332]. Storytelling is also a good skill to have in this process. Be detailed in describing the user's action when noting down the flow of the experience. Note down who performs the action, the exact steps of the action, and what is being affected by the action [334].

### B. PLATFORM CONSIDERATIONS

This section will cover some considerations to keep in mind when choosing the hardware and software platform for building an AR application.

#### 1) HARDWARE CONSIDERATIONS

In addition to the listed specifications or features of hardware, it is also necessary to consider factors beyond them. The technologies incorporated in a device could have unique traits or limitations that are not listed up front. A device with better specifications than another could still end up being worse in practice. Consider display technologies. Since OST displays generally combine the light of the display with the environment, it affects the quality of the visuals. For example, in the HoloLens 2, this means that color must be carefully chosen for the best experience. A color like black cannot be rendered with its additive display and must instead be mimicked with another color. The rendering is also affected by the user's environment and lighting conditions [335]. As a result, Microsoft has published design guidelines for best practices and considerations when displaying content on the HoloLens 2 [336]. Another example of traits and limitations would be the device's sensors. While the device may have the sensors for a function, it is also necessary to consider their range, precision, and accuracy. Take the HoloLens 2 once more. While it supports hand tracking, gesture recognition is limited to a "gesture frame" in front of the user [337]. The HoloLens also supports eye-tracking, which returns the predicted eye gaze. However, the result is approximately within 1.5 degrees in visual angle from the actual viewed

target. Thus, the precision of the selection can be affected at longer distances [34].

There can also be more factors related to the user experience of the device, such as comfort. The general comfort of wearing a headset or holding out a phone is tacit information best experienced firsthand. The distribution of weight with wearable devices like HMDs can also affect comfort. Two devices may weigh the same, but the device with better weight distribution and comfort will prove to be better. Comfort can also be affected by the type of display as well. Once again, with the HoloLens 2, Microsoft suggests that AR content be placed between 1.25 m and 5 m for maximum comfort and not be placed less than 40 cm from the user. To further avoid visual discomfort and fatigue, it is also recommended to minimize viewing content that moves between various depths. The same applies to rapidly switching focus between content placed near and afar [338]. Other device features that can affect the user experience include battery life (and uptime), device temperature, and breathability of cushioning materials.

#### 2) SOFTWARE CONSIDERATIONS

There are five main factors to consider when choosing a software platform for developing AR applications. Since these factors are more or less general for software development, we list them without elaboration for brevity:

- Feature set
- Supported platforms
- Conditions for distribution (licensing and pricing)
- Ease of development (documentation)
- Community size and support

### C. SOFTWARE DEVELOPMENT

In this section, we cover some guidelines for development and the importance of testing (and iteration) for development. This is followed by guidelines for creating 3D content for AR applications.

#### 1) DEVELOPMENT

When moving into development, there are a few things to keep in mind. Try to embrace standards where possible, as they will likely stay relevant regardless of how the technology evolves [324]. That being said, there can be multiple competitive standards, like various file formats for assets. In such cases, obtaining assets in a file format that can be converted into any other required file format is best. For example, glTF is a standard for 3D assets on the web, while ARKit utilizes USDZ. By obtaining assets in a file format like FBX, converting them into either glTF or USDZ will be possible [339]. Furthermore, using modular designs can allow for the reuse of components and speed up development. It can also allow for easy replacement or modification should changes be required in the future [324]. This is part of the idea behind toolkits like MRTK, as they provide common controls and behavior you can expect to see and use in an AR app [340].

#### 2) TESTING

Iteration and testing are crucial steps to identify and correct issues with the application [324]. The goal of testing is to stress the application by going outside the comfort zone. Simulation tools like the HoloLens [341] or Unity MARS [250] can be used to test the application without deploying it to a real device. While this can save time, it is important to remember that there is no perfect substitute for real-world testing. AR applications can be deployed in a wide variety of environments and should be tested as such. Be sure to test a variety of environments with different variables such as lighting and surfaces [324], [341]. Additionally, try testing with different users. This can cause unexpected behaviors, uncovering unforeseen aspects of the experience [341]. Utilize this opportunity also to test the accessibility of the experience where possible [324]. Iteration is important as there can be issues that will not show up during physical prototyping. For instance, it can be difficult to perceive depth in 2D due to the limited FOV, so placing objects at the right distance is crucial [132]. For HMDs, remember that the HVS has properties that can affect the quality of the experience [338]. After deploying the application, you can continue iterating based on user feedback [324].

#### 3) 3D CONTENT GUIDELINES

The first thing to be aware of when designing content for AR is the coordinate system used by a 3D application. All 3D applications use the Cartesian coordinate system. The position and orientation of a virtual object are defined along three perpendicular axes: X, Y, and Z. However, the applications do not follow the same conventions between them. For example, it is common for different applications to use either Y or Z to represent the direction ''up''. The unit of measurement used can also vary. 3D content creation tools like Blender allow users to set the unit system and its scale. When moving between applications, be sure to identify and pay attention to the differences in coordinate systems.

Unlike physical objects, virtual objects can adjust their scale on the fly to be larger or smaller. However, adjusting a virtual object's scale during an experience can have some unintended effects on the viewer. Consider the scenario where an object is scaled while remaining in place. This can make it seem like the object has been placed closer or further away than it is as it grows larger or smaller. If the user knows the size this virtual object is supposed to represent in the real world, this can lead to conflicting visual cues [329]. In general, it would be best to avoid changing scale specifically to mimic changes in distance (and vice versa).

Materials are essential to achieving the final look of 3D models. To create more realistic-looking assets, utilize the Physically Based Rendering (PBR) workflow for materials.

The PBR workflow generally involves combining multiple textures to create the final look of the object [342]. The following are the common textures utilized in a PBR workflow:

- Diffuse Map: This texture is used to determine the base color that needs to be rendered for the material.
- Normal Map: This texture is used to provide the appearance of more detailed geometry without the need for detailed geometry. This helps keep detail while reducing the need to have a complex 3D model at run time.
- Roughness Map: This texture is used to map how rough each area of a material should appear.
- Metallic Map: This texture is used to map how metallic each area of a material should appear.

To help correctly map these 2D textures onto a 3D object, a UV map texture is utilized. It breaks down the 3D object into its component flat 2D surfaces to determine which part of a 2D texture is applied to which face and how. Many additions to a material may be subtle, but they add up. Simple materials will only provide simple-looking objects. Utilize visual noise and its randomness to add more detail to the objects and break up repeating patterns. While higher-resolution textures can contain more detail, they will impact streaming and loading times. It is best to limit the resolution to improve performance [343].

In addition to the above textures, it may also be possible to improve the appearance of the object by adding more lighting details. Ambient Occlusion is a rendering technique used to calculate how exposed each part of an object is to the ambient lighting. By ''baking'' in an ambient occlusion map, it is possible to add shadows to less exposed regions of an object [342]. This helps add and accentuate detail, such as the crevices between the cushions of a couch. Baking is simply a term for making something permanent. By baking in an ambient occlusion map, we can add the appearance of ambient occlusion without having to run an expensive rendering process every time. Borrowing this philosophy, one can also fake shadows with a shadow plane, which is simply a plane containing only baked shadows [342]. Further utilizing AR features such as light estimation, environment probes, and occlusion will help immensely with blending virtual objects into the real world. One should be aware that rendering engines have differences between them in how to render. As such, the same object can look different in different engines.

### D. REGISTRATION

As discussed before, registration is a fundamental pillar of AR. In general, registration is the first step an AR application would need to perform. If the tracking system is part of a portable AR device, then the user will need to be a part of the registration process for the system to get its bearings and understand the environment. This initial registration step/experience needs to keep two principles in mind. First, there should be clear communication between the user and the system. The user must be kept aware if the application requirements are met. If not, then the user must be made aware of the issue and given possible resolutions with guidance. Second, strike a balance between efficiency and reliability. If the system can perform registration reliably without user assistance, that is optimal. User effort should only be utilized if necessary [341]. To design the initial registration experience, answer the following questions:

- What does the system need to find in the world?
- What information will the system already have?
- What actions can the system perform without user assistance?
- What actions will the user need to perform?
- What feedback will the user require?

The quality of registration/tracking (currently) is dependent on the user's environment or, more accurately, the environment perceived by the sensors. Limitations of the environment can affect the perception and degrade tracking quality and stability. Manipulating the environment to induce good and stable tracking is more of an art than a science [344]. A few examples of environmental limitations hindering registration are [326]:

- Flat surfaces without texture (e.g., A solid color desk)
- Dimly lit environments
- Extremely bright environments
- Transparent or reflective surfaces (e.g., Glass)
- Dynamic or moving surfaces (e.g., Flowing Water)

If the device cannot track the world, the device can only use an attached frame of reference to have device-locked AR content. Particularly with HMDs, this reference frame can be used to place content to alert the user about the issue [136].

For visual sensors to track the environment, they need to obtain enough visual information about the world. Sensors obstructed by hair or hands are an easy example of sensors lacking information [136], [344]. Cameras can also struggle to focus on real-world objects close to them, similar to human eyes. Therefore, always maintain a minimum distance [344]. When it comes to environmental factors, lighting affects the amount of information a sensor can get. If the environment is too bright, it saturates the cameras. If the environment is too dark, the cameras cannot see anything. Areas with contrasting lighting will make the camera adjust between bright and dim areas. This can cause the device to think a change in illumination equates to a change in location. Since outdoor lighting can vary across time, it can lead to inconsistent results [344].

Modern AR registration systems often rely on tracking features in the environment. Even with good lighting, there needs to be enough detail and variety in the environment for good tracking [344]. Reflective surfaces are a challenge beyond just lighting because the device can track reflected details on the surface. These details can change with the viewing angle, causing the system to lose tracking [344]. Since vertical surfaces often reflect light and are painted in

a single color, it makes them harder to detect than horizontal surfaces [132]. Having more detail is not limited to quantity but quality as well. If there are two identical areas in an environment, the system may get confused between them and mark them as the same location. This is an identified problem with the HoloLens 2. Identical areas can cause AR content to appear in different locations from where it was placed. It can also break tracking as the internal representation of the environment gets corrupted [136]. Since this makes the system think it is in two places at once, Microsoft calls this phenomenon ''wormholes'' [344]. Wormholes can be resolved by adding unique detail to the environment to distinguish between areas [344]. One very easy solution is to use masking tape to create unique and non-repetitive patterns along the surfaces of a space. There is also a simple test to know if there is enough detail in the environment and if it is unique enough: Can you uniquely locate yourself in the space if you only saw a small amount of the scene? If the user cannot locate themselves from visual cues, then the system will most likely struggle as well [344]. Motion and dynamic changes in the environment can also affect the tracking. As objects move around, there is nothing stable enough for the tracking system to use as a [136], [344]. Changes in the environment over a longer period can also affect tracking data [136]. Specifically, with spatial mapping, there can be ''ghosts'' of things that are no longer present. The tracking needs to be redone to obtain the most up-to-date information [344].

If tracking issues persist and the system is unable to correct them, allowing the user to reset the session is a good choice [136].

### E. DISPLAY

In this section, we will discuss possible modes of interaction and, thus, the possible interfaces for an AR application. While this has more to do with input than with display, the UI cannot be determined without first narrowing down the mode of interaction.

#### 1) INTERACTION AND INTERFACE

The interactions, controls, and interface are defined by the interaction model used with the experience. The interaction model chosen will depend on the type of experience that is to be delivered and the type of devices they will run on. For HMDs, Microsoft has identified three primary multimodal interaction models (see Table 8). For handhelds, virtual content can be divided into five interaction models based on three aspects of interaction [345]. We have made a few modifications to this classification. We have modified the names and definitions of the three aspects of interaction by [345] to the following:

- Content type: 2D or 3D.
- Location: The virtual content is placed on the screen (Screen Space) or in the real world (World Space).
- Dynamicity: The content is either fixed in a specified location (static) or can move around (dynamic). It can be locked or tied to other constraints.

We then reduced the number of models that can be created from this classification to the first four by [345]:

- 2D Static in Screen Space: 2D elements fixed in place on a screen.
- 2D Locked in World Space: 2D elements placed in the world. Usually locked to another object or point.
- 2D Dynamic in Screen Space: 2D elements that can be moved around on a screen. e.g., Targeting reticle for 3D object selection.
- 3D Flexible in World Space: 3D objects anchored or placed in the real world. They can be manipulated by the user if provided the ability.

The fifth model defined by [345] is ''dynamic 3D and proportionate in space''. It is defined as a model to help users to ''see an object in an actual environment with lighting and measurement considerations'' [345]. We ignore this model as the type of interaction model here is no different from ''3D flexible in world space''. Thus, we are left with four interaction models for handhelds (see Table 7. While touch is the primary mode of interaction for handhelds, the gestures and interaction systems can be different for the four groups.

The different interaction models can be combined if needed. However, this does come at the cost of increasing the complexity for the user [346]. One way to keep the controls simple and easy for the user is to make them intuitive. Users should be able to interact with the system without having to think about it as if the user and system are in sync [34], [347]. Using familiar controls is one way to make things more intuitive. We can rely on the user's prior knowledge, and the user does not need to learn a new set of controls [347]. Another way is to utilize ''implicit actions'', where the system automatically takes actions by gauging the context and user intent. Two examples covered previously are attentive holograms and eye-gaze-based auto-scrolling. Another example would be smart notifications that automatically vanish once the user is done reading (by utilizing eye-tracking) [34].

#### 2) UI

Two kinds of displays will be considered in this section. There are differences in designing UI when moving from a flat screen to an HMD. An HMD is inherently meant for a 3D context, while a screen is intended for 2D. The guidelines here will specify screen or HMD if they are limited to a display type. Attempts can be made to apply guidelines to a different display, but it needs to account for the traits of this display.

When designing UI for handhelds, try and support both orientations: portrait and landscape. Supporting both can increase user comfort and immersion. To avoid jarring transitions, try rotating the UI to match an orientation and avoid cutting the camera feed. Sometimes, different UI layouts may be needed for each orientation. Otherwise,

**TABLE 7.** Interaction models for handhelds.

| Interaction Model | Content Type | Location | Dynamicity |
|---|---|---|---|
| 2D Static in Screen Space | 2D | Screen Space | Static |
| 2D Locked in World Space | 2D | World Space | Locked |
| 2D Dynamic in Screen Space | 2D | Screen Space | Dynamic |
| 3D Flexible in World Space | 3D | World Space | Flexible |

**TABLE 8.** Multimodal interaction models for HMDs [346].

| Model | Example Scenarios | Fit |
|---|---|---|
| Hands and Motion Controllers | 3D spatial experiences, such as spatial layout and design, content manipulation, or simulation. | Great for new users coupled with voice, eye tracking, or head gaze. Low learning curve. Consistent UX across hand tracking and 6DOF controllers. |
| Hands-free | Contextual experiences where a user's hands are occupied, such as on-the-job learning and maintenance. | Some learning required. If hands are unavailable, the device pairs well with voice and natural language. |
| Gaze and Commit | Click-through experiences, for example, 3D presentations and demos. | Requires training on HMDs but not on mobile. Best for accessible controllers. |

users will have to adjust their hold of the device to interact with the UI. Changing the orientation may also change camera positioning. This could affect depth sensing, spatial awareness, and/or accurate surface measurements [347].

If additional information or controls are necessary, these UI elements can be directly displayed on a screen, like a HUD, separate from the AR experience (2D Static in Screen Space). Since they are locked to the screen, they will always be visible to users, unlike free floating content [135]. Direct object interactions (3D Flexible in World Space) are more intuitive and immersive than indirect controls. As such, they should be used when possible [135]. There are exceptions. For instance, persistent controls should consider utilizing indirect controls displayed directly on the screen [135]. Generally, fixed UI elements are placed at the top and/or bottom of the screen. This layout lets users focus on their AR experience [348]. If any 2D UI elements are placed in the world space (2D Locked in World Space), make them rotate to face the user unless otherwise required [135].

Cluttering the screen with controls and information can diminish the immersion [135]. Persistent 2D overlays are a constant reminder that the world the user looks at isn't entirely real. Alerts and notifications can also distract the user from the experience [347]. The screen should be devoted to displaying the AR session as much as possible. Limit onscreen controls to those used frequently or needing quick access (e.g., a camera shutter button) [139], [347]. Alternatively, utilizing translucency with UI can avoid blocking the underlying scene completely [135]. Sudden pop-ups, quick transitions, and fullscreen takeovers should be avoided, especially if the user does not initiate them. They could jolt the user and be unpleasant or shocking [139], [347]. If a device is not solely meant for AR, then the device likely needs to transition into the AR session. Keep this transition as seamless as possible. Give the user the control to trigger the AR session, as it is less jarring when they are in control. Utilizing animations such as fading can also help smoothen the transition [138]. Once inside the experience, try to maintain the continuity of the experience for immersion. Avoid taking the user out of the experience too often for other tasks. Try to keep all functions within AR as much as possible [135], [347].

The guidelines covered for handhelds could be adapted to HMDs. However, care should be taken as a 1:1 correspondence is not always possible or preferable. As an example, consider HUDs. HMDs can replicate on-screen UI elements by providing a head-locked HUD. While HUDs on a normal screen can keep the user informed without intruding, with HMDs they can be distracting and a cause for discomfort [136], [338]. For HMDs, it is best to avoid implementing 1:1 HUD rotation and translation based on head motions [338]. One solution is to lock the content to the user's body rather than their head [136], [338]. A good implementation would be a HUD that moves immediately with the user but only rotates with the user after a threshold is reached. The HUD can then reorient itself to provide a better angle to the user [338].

Using the right font and size is necessary to maintain text legibility and eye comfort. The limitations of hardware can be one factor affecting this. With displays like Hololens, thin fonts can vibrate and cause legibility issues [338], [349]. Legibility is also affected by the distance of the text to the user [349]. Besides the font type and size, it can be made more legible by utilizing a background that contrasts with the text. This can make it easier to distinguish the text from the environment [349]. To maintain consistency, avoid using more than two fonts in any single context [349]. Should there be text information, create and maintain a hierarchy for the information by using different text sizes to distinguish between them [349].

### F. INPUT

In this section, we discuss object placement and interaction. A discussion on input systems and guidelines for specific input modalities follows this.

#### 1) OBJECT PLACEMENT AND INTERACTION

The information in this topic is primarily meant for handhelds but can be reasonably used with HMDs with appropriate modifications. The primary difference would be that users can directly interact with the scene with HMDs, while the interactions are indirect with handhelds.

Before users can begin interacting with AR content, they must first be placed in the world. Objects can be placed into the scene automatically or manually [334]. With automatic placement, the app can start placing objects immediately once a surface is detected. This is preferable when [132]:

- there is minimal or no interaction
- objects do not need precise placement
- the AR experience must start immediately

With manual placement, users must manually place the objects into the scene. It is best-suited for [132]:

- a very interactive experience
- precise object placement

There are a few ways to implement manual placement. Users can simply tap on the screen to make the object appear in the scene. On handhelds, tapping is a very natural gesture for the user. Tapping is an ideal method when [132]:

- the object needs no adjustment or resizing before placement.
- the object needs quick placement

Another method is to let users drag the object into their scene after selecting it. It is important to let users know that this gesture is possible and how they can place down objects once picked up. Dragging works best when the object needs [132]:

- to be transformed or adjusted
- precise placement

Users can be provided a gallery of objects for selection. They can grab an object from the gallery and use the methods above to place them into the scene [139].

Placing objects directly onto the surroundings will require the surfaces to be tracked. It is necessary to communicate these tracked surfaces to the user so they can know where an object can be placed. A visual indicator can be utilized for this. By placing a visual indicator on a tracked surface, we can show the user exactly where an object can be placed [132], [135], [334]. Indicators can range in their complexity from simple images to 3D animations, depending on the need. They are not meant to be persistent and should disappear when the object has been placed [348]. Indicators can be implemented in different ways:

- A part of the tracked surface can be visualized
- Visualizing an object's shadow
- Placing another object (e.g., planes) on a tracked surface

An indicator can have more uses beyond just visualizing the position for the user. By aligning the indicator with the surface, we can convey the angle of the surface (and thus the angle at which the object will be placed). We can also convey the scale or size of the object to be placed by displaying it with the indicator. Animations can be used to transition from the indicator to the placed object to make it more interesting [334].

A few more quality-of-life features can be implemented to improve the experience. When moving objects for placement, set a maximum default distance where it can be placed. This ensures users do not place it too far away at an uncomfortable distance [132], [330]. Limits should also be placed on scaling [330]. If a user makes the object too large or too small, it will make it difficult for them to handle said objects. These limits for translation and scale are not as important to rotation. Particularly in utility apps, users are allowed to manipulate the object to view it from any angle [139]. Objects can be made to snap to make placement easier. Snapping will make the objects behave as if they are magnetic, forcing themselves into a specific point if they approach the general area. Objects could snap to other objects, tracked surfaces, or particular points on a tracked surface [334]. Scene understanding, such as plane classification, can fine-tune object placement. For example, if an object is only meant to be placed on a table, users are limited to placing the objects on surfaces classified as tables [135]. Once an object is placed, anchors can be utilized to keep them in place as the user moves around [132].

Small or faraway objects can be a challenge for users to select. In such cases, users should be given means for more precise selection, like reticles or raycasts [139]. Even then, precise selection may still prove difficult. In such cases, provide affordances to the user. Considering user proximity can provide leniency and make nearby objects react even if they are not directly selected [135], [330]. Making the target area for selection bigger than the objects is an alternative to proximity [139], [330].

Since object placement is reliant on tracking, tracking issues can bleed into objects placed in the world. A few things can be done to negate the effect of such issues. It is best to avoid aligning objects precisely with the edges of a tracked surface [135]. AR systems typically recalculate the frame of reference as they adapt to the environment [136]. The boundary of the tracked surface can easily change as more data is acquired. Thus, trying to place an object near a boundary is often a futile endeavor [135]. If tracking data is poor or not fully available, it is best to let the user place the object down regardless. Utilize what little information is available to approximate the object's place in the world. It is better to be responsive without precision than to be completely unavailable. The objects can be subtly nudged into the correct spot as more data becomes available [135]. Finally, when tracking is lost, make the application wait a few moments before making objects tied to it disappear. This is to help prevent the effect of flickering as tracking gets lost and found [135].

#### 2) INPUT SYSTEM GUIDELINES

When creating AR applications, consider incorporating multiple forms of input. This provides a few benefits to the experience. First, it provides a fallback should an input system be unavailable. Input systems may fail to operate

correctly in certain scenarios or may even be disabled for reasons such as privacy. In such cases, having a fallback will allow the experience to continue. Second, having a fallback helps improve accessibility. Some users may find it difficult to use a specific form of input. Alternate modes of input will give them a better chance to engage in the experience. Third, combining various input modalities will cover one modality's weakness with another's strength. This is especially useful if some form of input is finicky and can help make the overall experience smoother. For example, the user's proximity to virtual content can be combined with other inputs. This can be used to ensure the user only interacts with what they want to interact with. This also leads to the idea of ''affordances'', which is to reduce the precision required with any one input to provide more leniency to the user in their interaction [346].

This paper has discussed various input modalities available for AR (see Section IV-C: Input under Enabling Technologies). Below, we discuss some guidelines for a few inputs in particular.

#### 3) TOUCHSCREEN GESTURES AND HAND TRACKING

We will discuss touchscreen gestures together with hand tracking as they both involve the use of gestures and as such, share similarities. Touchscreen gestures are primarily used for directly interacting with 3D virtual content, similar to hand tracking. Both are utilized for four main interactions with 3D content:

- Selection
- Translation (moving the object)
- Rotation
- Scaling

In general, when creating gestures for interaction, try to avoid creating custom gestures. Custom gestures can add to the application's learning curve and create confusion when switching between them. Thus, it is best to stick to commonly used gestures for a known interaction [135], [324], [346]. Furthermore, the more instinctual the interactions feel, the smaller the learning curve will be [346]. Ensure that the system can distinguish between similar gestures through testing [135].

With these input modalities, the size of the object and its distance from the user will determine how users interact with them [346]. When it comes to on-screen/touch gestures, there are common gestures for object manipulation. Some require accommodating two-finger gestures commonly used for rotating and scaling an object. With rotation, it is possible to use one hand (index and thumb) or two hands (two thumbs) [330]. Scaling can be achieved with a simple pinch gesture, similar to zooming on a touch display [330]. Rotation can also be done with one finger, and it is good practice to support both options. However, this can introduce conflicts with other gestures [330]:

- To avoid conflicts with scaling, limit 2-finger rotation to when both fingers rotate in opposite directions on a horizontal axis.
- To avoid conflicts with translation, ensure a 1-finger rotation is limited to when the swipe gesture starts off the selected object.

While AR is 3D, touch-screen gestures are inherently 2D. Interactions can be simplified to accommodate this by limiting the axis or axes of motion. Consider the following examples [135]:

- Limiting movement to the 2D plane on which the object rests.
- Limiting object rotation to a single axis.

Another limitation that arises from the nature of a touch-screen is the difficulty in selecting small objects. The objects may appear small due to their actual size or distance from the user. Nevertheless, it can be difficult to precisely select the object in such cases, as a finger can be larger than the target and obscure the object from the user's view. There are two remedies available in this case:

- Increase the area/target size of the object to be selected, reducing the precision needed for selection (affordance).
- Increase the user's precision by utilizing ray casts for precise selection.

Moving on to hand-tracking interactions, at the simplest level, they involve mid-air hand gestures. By mimicking the touchscreen gestures through hands, they can be converted for use with hand-tracking [346]. A key difference, however, is that hand tracking does not face the limitations of touch gestures as it is not limited to 2D. Instead, we are introduced to a different set of challenges. On a touch screen, users can interact with any object that is visible on the screen. However, with hand tracking, users need to reach the object first. Since it is now possible for the hands to accidentally interact with nearby physical objects, it may be necessary to provide some affordances to the user. Users will also struggle with interacting with far-away objects. In such cases, ray casts can be used to extend the user's reach [346].

#### 4) VOICE INPUT

Voice input is a powerful input modality. Speech is an extension of thought, and thus, voice input allows the communication of intent. Knowing intent can simplify tasks and make the system more adept at responding to the user, providing the following advantages:

- Cognitive load is reduced as it is intuitive and easy to learn and remember.
- Effort is minimized by making tasks fluid and effortless.
- Time required to complete a task is reduced.

There are many examples of these advantages. Voice input can simplify menu traversal. Users can simply state the menu they wish to see and jump straight to it instead of navigating the menu hierarchy manually. Dictation can be utilized when virtual keyboards are hard to use. Voice input can also be used in conjunction with other modes of input. This can allow the users to multi-task. It can also enhance other modes of input, such as targeting an object with gaze and then using voice to declare intent [350].

The primary challenge with voice input is the reliability of detection. It is possible for the system to incorrectly hear and misinterpret a command [350]. The system needs to accommodate various accents and dialects that users can have [350]. The system will also struggle when dealing with unique or unknown words not in its vocabulary. Challenges with voice input are not limited to the technical realm. Voice input can be undesirable in shared spaces. It can be awkward to be seen talking to oneself in public. Privacy is limited as others can listen in. It can also disturb other users in the space. Finally, most systems currently utilize a library of possible voice commands. Users cannot naturally converse with the system and, thus, must learn the library of commands. This can add to the cognitive load instead of reducing it [350].

There are a few best practices that can be followed to improve the voice input experience. For one, keep commands concise and avoid similar sounding commands to maintain clarity [324], [350]. This will keep things simple and avoid confusion for both the user and the system. Limit commands to simple vocabulary where possible, as this reduces the load on the user to learn new terminology. Keeping the behavior of the commands consistent between various contexts will also reduce confusion for the user [350]. Voice input can struggle with continuous input control, such as volume adjustment or object manipulation [350]. But this can be alleviated to some extent by specifying the exact amount needed, instead of subjective words like ''louder'' or ''bigger''. Adequate feedback must be provided to let users know that their commands are being recognized [324]. Constraining voice commands to non-destructive actions will make it easier to revert changes in the case of any misinterpretation. Finally, test the system with different accents to ensure the detection is accurate [350].

#### 5) GAZE TRACKING

We have discussed utilizing gaze tracking to obtain directional information (see Section IV-C3: Gaze Tracking under Enabling Technologies: Input). However, gaze tracking can provide another form of information. By processing the duration of the gaze on a particular object, we can obtain ''attention''. Attention information can be utilized to gauge interest or obtain intent within the application [324], [328]. For instance, participants' gaze in a multi-user AR session can be visualized. This will allow other users to see what other participants are looking at. This can be useful for collaboration in a session [34]. Attention information can also be immensely helpful outside the application for debugging and improving the application. Attention information can be used to create heat maps. With such a heatmap, we can identify where users spend the most time [34] or if users have completely missed key objects or interactions [328]. This can streamline the application by identifying what is important and removing the rest. In training and productivity applications, attention information can be used to find bottlenecks to optimize the flow of tasks. This information can also be utilized in product design by identifying what aspects of the product a user finds interesting [34].

While eye and head tracking have been discussed together, they have fundamental differences. The specific features can make one more suitable than the other depending on the context. For example, consider the differences from a physical/biological perspective. Turning one's head is slow compared to the eyes. It can also get fatiguing with repeated motion. Eyes may allow faster movements with less effort but are more difficult to track. Eye movements are also sporadic and not smooth [34]. This makes eye tracking more suitable for interacting with large objects. It is not recommended for small targets or scenarios needing precision [34]. Head tracking tends to be smoother and more reliable and, therefore, a solid choice for most objects.

Now, let us consider the technical features. Head tracking can essentially fall under registration for HMDs (and handhelds). Registration is an essential component of AR. Therefore, the foundation for head tracking is always implemented. Eye tracking, on the other hand, will require dedicated sensors and will be more complex to implement. Additionally, eye tracking typically involves a calibration step for proper operation. This step can fail for a variety of reasons [34]:

- Incorrectly performing the calibration.
- System being unable to function correctly with corrective eyewear
- System being unable to accommodate certain eye conditions of the user
- External factors such as smudges on lenses of eyewear, lighting conditions, and occlusion of the eye (e.g., by the user's hair)

Eye tracking also requires collecting sensitive user data and therefore requires user permission [34], [324]. For these reasons, head tracking can be the fallback or alternative to eye-tracking [34]. However, head tracking is never a complete substitute for eye tracking. This is because head tracking does not provide the user's real gaze as the user could be looking somewhere else with their eyes [34]. For this reason, it is preferable to use eye tracking overhead tracking where possible.

When utilizing gaze tracking for interactions, a few practices must be followed. Avoid cluttering the visual space to provide clarity for the user [324], [328]. Similarly, it is important to provide feedback if gaze input is being utilized, and it should be kept subtle to avoid overwhelming the user [324]. The techniques for feedback can differ according to the tracking type. For example, a cursor can be utilized with head tracking to show the user where they are looking. The cursor should be smaller than the objects to avoid distracting the user from the actual content [34]. With eye tracking, eyes have sporadic movement, making the cursor too distracting. In this case, subtle visual highlights or animations can be used to create ''attentive'' objects. They can indicate themselves to the user by reacting to the user's gaze [34]. An example

would be an object that floats or changes color when gazed at. If timers are used with gaze for interaction, they should be carefully fine-tuned. Keeping them too short will make accidental selections more common. On the other hand, keeping it too long can make the interaction inefficient and cause user fatigue [34].

Finally, maintain user comfort. To avoid straining the user, excessive movements should be avoided with gaze tracking. A fixed gaze for long periods can cause strain [324]. When it comes to head tracking, moving the head can add strain and fatigue to the neck. As such, it is best to avoid the following movements:

- Gaze more than $10^{o}$ above or $60^{o}$ below the horizon (vertical motion)
- Neck rotations (horizontally) more than 45-degrees off-center

The optimal angle for the head (neutral/resting) is considered between $10^{o}$-$20^{o}$ below the horizon. It is normal for the head, especially during activities, to tilt downward slightly from the resting angle [338].

### G. GENERAL AR APPLICATION DESIGN: BEST PRACTICES

This section covers some best practices to follow when designing an AR application. They have been divided into four categories.

#### 1) ENVIRONMENT AND MOVEMENT

As AR is registered in 3D, the environment is a significant component of the experience. Making content move between various surfaces and interacting with them makes the experience seem more immersive and showcases the environment [139]. It is also possible to have visual effects that interact with the environment [334]. Having the user move in the environment to observe content is another way to leverage this strength of AR. The user's movement will be defined by the type of experience. Here are some guidelines for user movement:

- Let users know what movements are possible.
- Guide them through the types and range of movement.
- Provide easy transitions from one pose/movement to another.
- Prioritize comfort. Avoid physically demanding, uncomfortable, or sudden movements [351].
- Easing users into movement allows them to adapt [135], [351].

There are various ways to encourage users to move around in their environment. Hiding virtual objects behind real or virtual objects can be used to reward and incentivize exploration [139], [351]. Large virtual objects can make people walk around to see it in its entirety [139]. Users can be encouraged to look around by placing objects towards the periphery [328]. Alternatively, audio and/or visual cues can be used to guide them to offscreen content [135], [138], [139], [328], [334]. By placing content offscreen, the environment can also be used to offset any limitations in FOV by placing content out of view but with indicating cues. It is important to ensure that these cues are not too distracting to the user. Using too many visual cues can clutter up the screen. With audio cues, to avoid distracting the user, use the following tips [138]:

- Avoid playing sounds simultaneously
- Add attenuation to moderate sound effects
- Set the audio to fade or stop if a user is not interacting with the object
- Allow users to manually turn off the audio for individual objects

If cues can be used to tell the user where to look, they can also be used to tell where they should not look. As the user is free to move around, they will be free to look inside objects. Cues like dimming the screen can tell the user that they are not meant to be looking inside [139].

Movement can be challenging when displaying a virtual experience larger than the physical space. Since the experience is larger than the physical space, the user cannot physically move around the virtual experience. In such cases, some form of virtual movement is required. Automatic movements and movements that go against the user's gravity can be disorientating. This is because the human vestibular system (for balance and orientation) can interact with the HVS. If visual cues from the virtual experience do not align with vestibular cues from the real world, it leads to motion sickness. Therefore, always keep the user in control of their movements. The following steps can be used to make a system for user navigation in a large scene [338]:

- select a point in the scene
- shrink the entire scene down to that point
- allow users to move the point close to their feet
- expand the scene again

#### 2) COMFORT, SAFETY, AND SECURITY

It is important to consider the comfort, safety, and security of the users for the continued use of the application. The first step would be to consider the devices' quirks. For example, with the HoloLens 2, objects should ideally be placed 2m away from the user to avoid eye discomfort. However, switching between distances causes more discomfort than an object placed close to the user [338]. Another example would be the device's weight, adding to the user's fatigue as the session continues. As the users get tired, they will want to move around less. If a user is not able to move around, give them an alternative way to use your app [351]. This can also improve the application's accessibility through an alternative mode of input [324]. For example, users can have muscle fatigue from holding their arms in the air for too long. Using alternate forms of input can provide the users with a break [135], [338], [352]. Avoiding long sessions and allowing the experience to be paused and restarted at the user's convenience will reduce problems from fatigue [352].

Users can get too engrossed with their experience to pay attention to the environment. This can be a problem if users

are encouraged to move during the experience, raising the probability of an accident. There are a few steps that can be taken to avoid incidents:

- Give users reminders to look around and check their surroundings [352].
- Avoid large and sudden movements by the user as they will not be aware of their surroundings when immersed [135].
- Do not make the user walk backward. They are more likely to trip when unaware of what is behind them [139], [352].
- Do not startle users as they may jump backward and trip [139].

If any of the movements listed above are required, ensure the user is safe before asking them to do so. Precautions must be taken if the users will be using AR in public. Moving in public while immersed in AR can be distracting and dangerous [326].

Security primarily comes into play when the user's privacy can be violated (see Section XV: Current Challenges in AR). User permissions have been one way to communicate to the user what information the system will need access to. To encourage user cooperation, clearly state why certain permissions are required and provide the relevance and benefits of each permission. Additionally, only ask for permissions when it is necessary to move forward in an experience [135], [347]. While the methods above are certainly valid, we would like to note that they are a bit amoral. Providing only the relevance and benefits does not apprise the users of the cost and risks of providing their data. Asking for permission to continue during the experience also encourages acceptance without further consideration. That being said, focusing on the negatives can cause users to avoid the application unnecessarily. It can be argued that the most important thing is to be transparent and capable of protecting the user's data should the application collect it.

#### 3) FEEDBACK AND GUIDANCE

Currently, for the vast majority of people, XR (let alone AR) is a completely new experience. Users need to be provided with appropriate guidance and feedback to navigate an AR experience. To avoid overloading the user with information, make the tutorial a part of the main experience. Show how to perform tasks as they come up instead of teaching everything at once. This will make it easier to link instructions and tips to the task at hand [347]. Avoid using technical terms like registration and tracking to be approachable to new users. Instead, use friendly, common, and conversational terms that most people will understand [135]. See Table 9 for some examples of how to provide technical information in a more approachable fashion.

The first step in an experience would be registration. To scan the environment properly, it may be necessary to show the users how to move the device, preferably with visual aids like animations [132], [135], [347]. When trying to coach or guide the user, hide unnecessary UI. This will help the user focus on the instructions [135]. When a surface has been detected, provide visual confirmation to the user and tell them what to do next. Avoid highlighting multiple surfaces at once. Only highlight the surface that the user is viewing or pointing at. If multiple surfaces exist, ensure they are visually distinct [132]. When tracking is lost, and the system needs to regain its bearings, it undergoes a process called ''relocalization''. Like registration, guide the users back to their original position and orientation when trying to relocalize a position [135]. Libraries and toolkits may provide pre-made assets for guiding the user through registration, avoiding the need for custom assets or to build assets from scratch [135].

Once the experience begins, make it easy to spot interactive objects by using visual highlights like glowing outlines. This can also prove useful in scenarios with multiple virtual objects to let the user know their selection [330]. As discussed before, visual and audio cues can be used to guide users to offscreen content. They can also be used to let the user know if they are looking the wrong way, like inside an object or in the wrong direction [334]. If there is more information that the user can know at any point, let them know it is available [135].

Since AR is primarily a visual experience, try to visually guide and provide feedback. Providing text instructions, especially for tacit information on motion, can take users out of the experience and make instructions hard to remember [347]. It is easier to understand and remember motion by seeing it than by reading about it. Similarly, it is better to use 3D hints for actions in 3D space than 2D hints. Only revert to 2D if people are not able to deal with the 3D context well [135]. For example, instead of providing text instructions for the direction of rotating an object, show the arrows in 3D. Since it is a visual medium, animation is also useful for providing feedback and guidance in AR. Try to think of it as a necessity instead of a luxury [333]. Visual cues can also leverage traits of the HVS to their advantage. For instance, the eyes are naturally drawn to motion and high-contrast areas. This can be used to control the user's attention [324]. Audio [135], [328], [333] as well as haptic cues [135] are alternatives to visual cues. Try not to go overboard when guiding the user's gaze to avoid overwhelming and limiting the users [324], [328].

#### 4) ERRORS AND ISSUES

When the application has errors or issues, communicate what went wrong and proceed to the resolution [132], [135], [347]. Focus on the resolution and avoid blaming the user [347]. Walk the user through correcting the issue and proceed one step at a time by giving simple and short tasks [132]. Use straightforward and friendly language when offering suggestions [135]. Fixes obvious to the developer will not be obvious to a naive user. Remember to use approachable terminology (see Table 9). If the errors or issues that pop up

**TABLE 9. Examples of approachable terminology [135].**

| Technical | Approachable |
|---|---|
| Unable to find a plane. Adjust tracking. | Unable to find a surface. Try moving to the side or repositioning your phone. |
| Surface detection takes too long. | Try moving around, turning on more lights, and making sure your phone is pointed at a sufficiently textured surface. |
| Tap a plane to anchor an object. | Tap a location to place the [name of the object to be placed]. |
| Insufficient features. | Try turning on more lights and moving around. |
| Excessive motion detected. | Try moving your phone more slowly. |

during an experience cannot be fixed, give them a way to reset the experience [135], [138]. This is particularly applicable to issues relating to tracking [135]. Tracking issues can also lead to flickering or other unwanted visual effects. It is best to hide virtual objects in such cases until the system can properly resolve tracking [135]. Last but not least, some issues just cannot be resolved. For instance, sometimes, there is no remaining fallback for input. Even if there is no resolution, it is important to let users know about it [34].

## XV. CURRENT CHALLENGES IN AR

Except for FOV, we do not list technological challenges in this section, even though they most certainly exist. With this paper covering multiple technologies without enough depth, we cannot do them all justice. Furthermore, technological challenges are ever-present, a point we address under ''Value''.

### A. VALUE

For AR to have mass adoption, it needs to have value. While value is subjective, we anticipate at least four factors that will affect it: function, usability, price, and appeal. These factors are not mutually exclusive and can influence each other. Function refers to what task the AR solution accomplishes. Advancements in technology can always improve the stated factors. However, there will always be technological challenges because there will always be constraints to overcome. One can always wait for better technology that arises from resolving these challenges. However, there is no guarantee that a complete solution will achieve the resolution. It could always be a tradeoff, e.g., remote computing for AR. While improvements are always appreciated, good software is needed more, particularly applications that provide value to the user [353]. Reference [35] reflects this sentiment, citing ''Killer MAR Applications'' as a challenge to be addressed. We address this point ourselves with the ''Litmus Test''. By focusing on applications, we focus on function and appeal and, to a degree, surpass issues with usability and price arising from technology. That being said, there is one additional task beyond just the application: marketing. It is necessary to make users aware that such value exists. If they do not know, then it might as well not exist. Marketing can also play into appeal. Immersive media as a concept has already existed even as far back as 2017 with 360° photos [353]. But it is Apple with their Apple Vision Pro claiming to be ''revolutionary'' by providing immersive media, and thus, will be the ones seen as ''revolutionaries.''

### B. FOV

Improvements in FOV can be a huge boon for AR. Generally, ''the larger the FOV, the more immersive the AR experience will be.'' Reference [31] With a small FOV, even small eye or head movements can make the digital content disappear from view. Thus, a small FOV can easily affect the immersion and realism of the experience as the difference between real and virtual becomes too obvious. This may also force the user into unnatural movement patterns as they struggle to keep the digital content in view. It is for this reason we have mentioned FOV over other technological challenges. We believe that with the current state of technology, in general, increasing the FOV will provide the biggest increase in AR's value by improving its usability. Many commercial AR solutions today do not come close to the typical horizontal FOV of human vision, which is 180°-200° when combining both eyes. With handhelds and other mobile devices with screens, the FOV is limited to the size of their screens, which tend to be small to begin with. With HMDs, OST HMDs, in particular, have much lower FOV than their VST counterparts. Finally, SAR solutions would be limited by their ability to project onto the environment. While strides are being made to provide better FOVs, the solutions must remain practical for mass adoption [31].

### C. PRIVACY

Privacy has been identified as a challenge for AR before [35], and it will be a greater one as AR becomes more widely adopted. The capabilities of AR systems make them (in theory) the ideal platform for mass surveillance. Most, if not all, AR systems will have a camera for registration. Therefore, AR applications require camera access at the very minimum to be functional. Many AR systems also incorporate microphones. This privilege can be abused to view the user's surroundings or listen to their conversations. Since these concerns were raised for other applications well before AR, obtaining camera and microphone access on phones already requires asking the user for their permission. The general trend in AR is to have a system capable of understanding its environment better. This can be seen in the advanced AR software features listed previously. However, these advanced features can encroach further into the user's privacy. Location tracking and face tracking can be abused to track the user. Eye tracking can be used to compromise biometric information. Spatial mapping can be used to map the user's surroundings. Advancements in AI for features like scene semantics can be utilized to correctly break down every

item in a user's environment, destroying privacy by providing a complete picture/breakdown of their private lives. Remote computing will put user data into third-party hands, thus making a third party responsible for the privacy and security of the data.

### D. XR FRAGMENTATION

We have already discussed XR fragmentation and its solutions under AR standards. However, we argue that XR fragmentation is not limited to the technology. Standardization is needed on other fronts. A common issue we would encounter would come down to terminology and definitions. Within the industry, it is not uncommon to consider MR and AR to be distinct despite AR falling under MR in the Reality-Virtuality Continuum. The naming of the various AR features can also be conflicting. Scene understanding, spatial mapping, scene mapping, and scene semantics are all terms thrown around that can mean different things under different libraries. Another example would be environment probes, which may or may not include light estimation depending on the context. Similar effects can also be seen in academic work. Reference [31] makes the argument that characteristics of AR alone do not lead to an AR experience, specifically stating the real and virtual need to be indistinguishable. While visual coherence is needed for a good experience, being indistinguishable is not necessary for AR. Standardization can help resolve such confusion and keep AR moving forward.

## XVI. FUTURE STEPS

The motivation of this paper was to provide a comprehensive introduction. However, owing to constraints, we had to sacrifice certain elements from the original scope of this paper. In this section, we discuss some of our shortcomings and, by extension, future steps for this research.

One glaring omission in this paper is any mention of digital twins. It is important to AR as an enhancing technology, and needs to be discussed in further detail.

Despite being an introduction, we lack a section on AR history. AR history needs to be identified and archived. Considering the players in this industry, it will be an arduous task. But it is important to understand why things are the way they are, e.g., the various variants of ARToolKit. We also wanted to include a survey of AR applications. Typically, applications for AR would be identified and grouped according to various fields, such as medicine or education. However, such surveys never go into the project's implementation and techniques. They would also limit themselves to academic papers when it comes to applications. AR is now accessible to the general public, and we must expand beyond academia to find the best applications. We believe a survey of applications beyond academia and a focus on implementation would be more useful because it would better identify AR applications that showcase the potential of the technology.

Our paper focuses primarily on visual AR, specifically for handhelds and HMDs. Our information is limited for SAR and AR for other senses. In addition, the list of hardware and software can be expanded. For example, the Varjo XR-4 is one HMD we are aware of but could not include in this paper. The section on hardware and software could also be expanded with hands-on reviews. This can provide comparisons on the viability of each solution.

In the section for design guidelines, the information was spread across multiple sources and even across different mediums (videos, apps, etc). Some sources are provided on a best-effort basis. Furthermore, the information in the guidelines section often gets split between handhelds and HMDs. It may be possible to combine the knowledge to create universal guidelines such as universal interaction models. The guidelines could also be expanded with a basic but more comprehensive introduction to 3D or video-game development, specifically for AR. While we have already covered a few topics under 3D content guidelines, we believe it is important to showcase some basics of 3D to understand better what can be achieved with AR.

Finally, the long-term health effects of AR is another interesting topic that needs to be explored. We have gone from our parents telling us not to be too close to the TV to strapping displays right next to our eyes. Regardless of the veracity of such claims, consequences to health should be an important factor in the long-term success of a technology. We could not find any mention of health effects in a survey on AR.

## XVII. CONCLUSION

In this paper, we aimed to provide a comprehensive introduction and start to the field of AR. We have provided an introduction to AR and covered the essential technologies that enable AR. We also covered additional technologies that could enhance AR further. This was followed by a discussion on commercially available AR hardware, in which we covered 18 HMDs (including those under ''Other Devices''). The seven primary HMDs and the five AR Glasses have Tables with their specifications for easy comparison (see Appendix). This was followed by a section on AR software features, where we provided descriptions and illustrations for features found in AR software. In the discussion on AR software, we have covered 26 software solutions for development, with an additional three listed by name, for a total of 29 software. Feature matrices were provided for easy comparison between 13 of the 29 software. We then covered guidelines for AR development, covering the complete flow of developing an AR application. Finally, we discussed current challenges in AR and future steps that could be taken to improve this paper.

## APPENDIX. TABLES

Tables 10 and 11 for AR HMDs and AR Glasses, respectively.

**TABLE 10.** Device specifications for AR HMDs.

| | Microsoft Hololens 2 | Magic Leap 2 | Meta Quest Pro | Meta Quest 3 | HTX Vive XR Elite | Varjo XR-3 | Vrgineers XTAL 3 Mixed Reality |
|---|---|---|---|---|---|---|---|
| **Device Type** | Standalone | Standalone | Standalone | Standalone | Standalone | Tethered | Tethered |
| **Display** | OST<br>1440 x 936 per eye<br>60Hz* | OST<br>1440 x 1760 per eye<br>120 Hz refresh rate | VST<br>1800 x 1920 per eye<br>90 Hz | VST<br>2064 x 2208 per eye<br>90Hz<br>(72Hz, 80Hz, 120Hz)* | VST<br>1920 x 1920 per eye<br>(3840 x 1920 total)<br>90 Hz | VST<br>Focus area (27°x27°):<br>1920 x 1920 per eye<br>Peripheral area:<br>2880 x 2720 per eye<br>90 Hz | VST<br>3864 × 2192 (45Hz) per eye<br>2232 × 2192 (75Hz) per eye<br>Foveated Rendering:<br>3840 × 2192 (90Hz) per eye |
| **FOV** | 43° Horizontal<br>29° Vertical<br>52° Diagonal | 44.6° Horizontal<br>53.6° Vertical<br>70° Diagonal | 106º Horizontal<br>96º Vertical | 110º Horizontal<br>96º Vertical | Up to 110º | 115° Horizontal | 175° Horizontal and 100° Vertical*<br>180° Horizontal and 90° Vertical<br>63° Horizontal and 38° Vertical* |
| **SoC** | Qualcomm<br>Snapdragon 850 | AMD<br>Quad Core Zen 2 | Qualcomm<br>Snapdragon XR2+ | Qualcomm<br>Snapdragon XR2<br>(Gen 2) | Qualcomm<br>Snapdragon XR2 | N/A (Tethered) | N/A (Tethered) |
| **RAM (GB)** | 4 | 16 | 12 | 8 | 12 | N/A (Tethered) | N/A (Tethered) |
| **Storage (GB)** | 64 | 256 | 256 | 128<br>512 (Alternate Config) | 128 | N/A (Tethered) | N/A (Tethered) |
| **Still (Camera)** | 8-MP | 12.6MP | Unknown | Unknown | 16 MP | Unknown | Unknown |
| **Video** | 1080p30 video | 4K at 30FPS<br>1920x1080 at 60FPS | Unknown | Unknown | Unknown | Unknown | Unknown |
| **Depth Sensing** | 1-MP ToF sensor | Depth Camera | 5 Sensors* | 2 RGB Cameras*<br>Depth Projector* | Depth sensor<br>Proximity sensor | LiDAR &<br>RGB fusion* | Unknown |
| **Audio** | 5 Microphone array<br>Built-in speakers<br>(Spatial Audio) | 2 Built-in<br>Stereo Speakers<br>4 Microphones | 4 Integrated speakers<br>(Spatial Audio)<br>3 Microphone Array<br>2 3.5mm audio jacks | Integrated speakers<br>(Spatial audio)<br>Microphone<br>3.5mm audio jack | Embedded Speakers<br>Dual Microphones | 3.5mm audio jack<br>(microphone support) | Unknown |
| **Orientation and Other Sensors** | Accelerometer<br>Gyroscope<br>Magnetometer | 4x IMU<br>3-axis Accelerometer<br>3-axis Gyroscope<br>2 3-axis Magnetometer<br>2 Altimeter | Unknown | Unknown | Accelerometer*<br>Gyroscope | Unknown | Unknown |
| **Hand Tracking** | Yes | Yes | Yes | Yes | Unknown | Yes* | Yes* |
| **Eye Tracking** | Yes | Yes | Yes | No | Unknown | Yes* | Yes* |
| **Face Tracking** | No | No | Yes | No | No | No | No |
| **Controllers** | No | Yes | Yes | Yes | Yes | No | No |
| **Wifi** | Wi-Fi 5 | Wifi 6 | Wi-Fi 6E | Wi-Fi 6E | Wi-Fi 6 + 6E | N/A (Tethered) | N/A (Tethered) |
| **Bluetooth** | Bluetooth 5.0 | Bluetooth 5.0 | Bluetooth 5.2 | Bluetooth 5.2* | Bluetooth 5.2 | N/A (Tethered) | N/A (Tethered) |
| **Weight** | 566g | 260g (HMD)<br>N/A (Compute Pack) | 722g | 515g | 273g<br>625g (with cradle)* | 594g<br>386g (Headband) | 700g (without headstrap) |
| **Battery Life** | 2–3 hours | 3.5 hours | 2.5 hours | 2.2 hours | 2 hours | N/A (Tethered) | N/A (Tethered) |
| **Price** | $3,500 | $3,299 | $999.99 | $499.99 (128GB)<br>$649.99 (512GB) | $1,099 | €6495* | $11,800 |

**TABLE 11.** Device specifications for AR glasses.

| Name | Google Glass Enterprise Edition 2 [354] | Epson Moverio BT-40S [355] | Nreal Light [356] | Nreal Air [357] | Vuzix M400 [358] |
|---|---|---|---|---|---|
| **Device Type** | Standalone | Standalone<br>(External Compute Pack) | Phone-powered AR | Phone-powered AR | Standalone AR |
| **Display** | OST<br>640x360 per-eye (single eye) | OST<br>1920x1080 per-eye<br>60Hz | OST<br>1920x1080 per-eye<br>60 Hz | OST<br>1920x1080 per-eye<br>60 Hz | VST*<br>Single OLED monocular |
| **FOV** | NA | 34° diagonal | 52° diagonal | 46° diagonal | 16° diagonal |
| **Camera** | 8MP, 1080p @ 30FPS | NA | 5MP | NA | 12.8 MP 4K @ 30 FPS |
| **Audio** | Integrated mono speaker<br>Microphone | Integrated speaker<br>Microphone<br>3.5mm audio jack | Integrated stereo speakers<br>Microphone | Integrated stereo speakers | Integrated mono speaker<br>Microphone |
| **Registration** | 3DOF Non-positional | 3DOF Non-positional | 6DOF Inside-out<br>(via 2 integrated cameras) | 3DOF Non-positional | 3DOF Non-positional |
| **Other tracking** | NA | NA | Hand Tracking | NA | NA |
| **OS** | Android 8.1 | Android | NA | NA | Android 9 |
| **Compute** | Qualcomm Snapdragon XR1<br>3 GB RAM<br>32GB Storage | Qualcomm Snapdragon XR1<br>4 GB RAM<br>64 GB Storage<br>SD card expansion | NA | NA | Qualcomm Snapdragon XR1<br>6 GB RAM<br>64 GB Storage |
| **Connectivity** | WiFi 802.11a/g/b/n/ac<br>Bluetooth 5.0 | USB 3.0 Type C<br>USB 2.0<br>WiFi 802.11 a/b/g/n/ac<br>Bluetooth 5.0 | USB Type-C | USB Type-C<br>Lightning | USB 3.1 Gen 2 Type-C<br>WiFi 802.11 a/b/g/n/ac<br>Bluetooth 5.0 BR/EDR/LE |
| **Weight** | 46 g<br>51 g (with headstrap) | 96 g | 106 g | 77 g | 182 g |
| **Battery Life** | 8 hours | Unknown<br>(3400 mAh capacity) | No battery<br>(powered by phone) | No battery<br>(powered by phone) | 2 hours<br>12 hours (external battery) |
| **Compliance** | IP53 | IPX2 (excluding USB port)<br>RoHS | None | None | P67, IEC60601-1-2:2014 |
| **Price** | $999 (Discontinued) | $940.00 | $499.00 | $400.00 | $1,800.00 |

## REFERENCES


[1] Y. Asham, M. H. Bakr, and A. Emadi, "Applications of augmented and virtual reality in electrical engineering education: A review," *IEEE Access*, vol. 11, pp. 134717–134738, 2023.

[2] M.-B. Ibáñez and C. Delgado-Kloos, "Augmented reality for STEM learning: A systematic review," *Comput. Educ.*, vol. 123, pp. 109–123, Aug. 2018.

[3] N. F. Saidin, N. D. Abd Halim, and N. Yahaya, "A review of research on augmented reality in education: Advantages and applications," *Int. Educ. Stud.*, vol. 8, no. 13, pp. 1–8, Jun. 2015.

[4] E. Z. Barsom, M. Graafland, and M. P. Schijven, "Systematic review on the effectiveness of augmented reality applications in medical training," *Surgical Endoscopy*, vol. 30, pp. 4174–4183, Sep. 2016.

[5] M. Eckert, J. S. Volmerg, and C. M. Friedrich, "Augmented reality in medicine: Systematic and bibliographic review," *JMIR mHealth uHealth*, vol. 7, no. 4, Apr. 2019, Art. no. e10967.

[6] H. A. A. M. Abas, F. B. A. Aziz, and R. Hasan, "Review of augmented reality applications in manufacturing engineering," *J. Adv. Res. Comput. Appl.*, vol. 5, pp. 11–16, Aug. 2016.

[7] S. Agarwal, "Review on application of augmented reality in civil engineering," in *Proc. Int. Conf. Inter Disciplinary Res. Eng. Technol.*, vol. 68, 2016, p. 71.

[8] W. Jia, J. Zhu, L. Xie, and C. Yu, "Review of the research on augmented reality maintenance assistant system of mechanical system," *J. Phys. Conf. Ser.*, vol. 1748, no. 6, Jan. 2021, Art. no. 062041.

[9] Z. Makhataeva and H. Varol, "Augmented reality for robotics: A review," *Robotics*, vol. 9, no. 2, p. 21, Apr. 2020.

[10] M. E. de Oliveira and C. G. Correa, "Virtual reality and augmented reality applications in agriculture: A literature review," in *Proc. 22nd Symp. Virtual Augmented Reality (SVR)*, Nov. 2020, pp. 1–9.

[11] M. A. Bretos, S. Ibáñez-Sánchez, and C. Orus, "Applying virtual reality and augmented reality to the tourism experience: A comparative literature review," *Spanish J. Marketing*, vol. 28, no. 3, pp. 287–309, Jul. 2024.

[12] Y. Zhou, J. Chen, and M. Wang, "A meta-analytic review on incorporating virtual and augmented reality in museum learning," *Educ. Res. Rev.*, vol. 36, Jun. 2022, Art. no. 100454.

[13] N. S. Matthie, N. A. Giordano, C. M. Jenerette, G. S. Magwood, S. L. Leslie, E. E. Northey, C. I. Webster, and S. Sil, "Use and efficacy of virtual, augmented, or mixed reality technology for chronic pain: A systematic review," *Pain Manage.*, vol. 12, no. 7, pp. 859–878, Oct. 2022.

[14] J. G. Kovoor, A. K. Gupta, and M. A. Gladman, "Validity and effectiveness of augmented reality in surgical education: A systematic review," *Surgery*, vol. 170, no. 1, pp. 88–98, Jul. 2021.

[15] K. A. Bölek, G. De Jong, and D. Henssen, "The effectiveness of the use of augmented reality in anatomy education: A systematic review and meta-analysis," *Sci. Rep.*, vol. 11, no. 1, p. 15292, Jul. 2021.

[16] R. Z. Ramli, W. Z. Wan Husin, A. M. S. Elaklouk, and N. S. Ashaari, "Augmented reality: A systematic review between usability and learning experience," *Interact. Learn. Environments*, vol. 1, pp. 1–17, Sep. 2023.

[17] A. Dey, M. Billinghurst, R. W. Lindeman, and J. E. Swan, "A systematic review of 10 years of augmented reality usability studies: 2005 to 2014," *Frontiers Robot. AI*, vol. 5, p. 37, Apr. 2018.

[18] K. C. Lim, A. Selamat, R. A. Alias, O. Krejcar, and H. Fujita, "Usability measures in mobile-based augmented reality learning applications: A systematic review," *Appl. Sci.*, vol. 9, no. 13, p. 2718, Jul. 2019.

[19] M. D. Gómez-Rios, M. Paredes-Velasco, R. D. Hernández-Beleño, and J. A. Fuentes-Pinargote, "Analysis of emotions in the use of augmented reality technologies in education: A systematic review," *Comput. Appl. Eng. Educ.*, vol. 31, no. 1, pp. 216–234, Jan. 2023.

[20] M. Akçayır and G. Akçayır, "Advantages and challenges associated with augmented reality for education: A systematic review of the literature," *Educ. Res. Rev.*, vol. 20, pp. 1–11, Feb. 2017.

[21] D. G. Molina Vargas, K. K. Vijayan, and O. J. Mork, "Augmented reality for future research opportunities and challenges in the shipbuilding industry: A literature review," *Proc. Manuf.*, vol. 45, pp. 497–503, Oct. 2020.

[22] E. Coronado, S. Itadera, and I. G. Ramirez-Alpizar, "Integrating virtual, mixed, and augmented reality to human–robot interaction applications using game engines: A brief review of accessible software tools and frameworks," *Appl. Sci.*, vol. 13, no. 3, p. 1292, Jan. 2023.

[23] M. Vásquez-Carbonell, "A systematic literature review of augmented reality in engineering education : Hardware, software, student motivation & development recommendations," *Digit. Educ. Rev.*, vol. 1, no. 41, pp. 249–267, Jul. 2022.

[24] R. T. Azuma, "A survey of augmented reality," *Presence, Teleoperators Virtual Environ.*, vol. 6, no. 4, pp. 355–385, 1997.

[25] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic, "Augmented reality technologies, systems and applications," *Multimedia Tools Appl.*, vol. 51, no. 1, pp. 341–377, 2011.

[26] C. Bermejo and P. Hui, "A survey on haptic technologies for mobile augmented reality," *ACM Comput. Surveys*, vol. 54, no. 9, pp. 1–35, Dec. 2022.

[27] D. Rumiłski, "An experimental study of spatial sound usefulness in searching and navigating through AR environments," *Virtual Reality*, vol. 19, nos. 3–4, pp. 223–233, Nov. 2015.

[28] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual-displays," *IEICE Trans. Inf. Syst.*, vol. 77, no. 12, pp. 1321–1329, 1994.

[29] U. Neumann and S. You, "Natural feature tracking for augmented reality," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 53–64, Mar. 1999.

[30] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: A hands-on survey," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 12, pp. 2633–2651, Dec. 2016.

[31] Y. Itoh, T. Langlotz, J. Sutton, and A. Plopski, "Towards indistinguishable augmented reality: A survey on optical see-through head-mounted displays," *ACM Comput. Surveys*, vol. 54, no. 6, pp. 1–36, Jul. 2022.

[32] A. Samini, K. L. Palmerius, and P. Ljung, "A review of current, complete augmented reality solutions," in *Proc. Int. Conf. Cyberworlds (CW)*, Sep. 2021, pp. 49–56.

[33] A. Ahmad, C. Migniot, and A. Dipanda, "Hand pose estimation and tracking in real and virtual interaction: A review," *Image Vis. Comput.*, vol. 89, pp. 35–49, Sep. 2019.

[34] *Eye Tracking Overview—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/eye-tracking

[35] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[36] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1160–1192, 2nd Quart., 2021.

[37] *Touch Haptic Device*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.3dsystems.com/haptics-devices/touch

[38] *Desktop 6D—Haption SA*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.haption.com/en/products-en/virtuose-6d-desktop-en.html

[39] *Novint's Falcon Haptic Device—Hapticshouse.com*. Accessed: Apr. 18, 2023. [Online]. Available: https://hapticshouse.com/pages/novints-falcon-haptic-device

[40] *Novint Technologies–Crunchbase Company Profile & Funding*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.crunchbase.com/organization/novint-technologies

[41] *Collections—Hapticshouse.com*. Accessed: Apr. 18, 2023. [Online]. Available: https://hapticshouse.com/collections

[42] *Senso Devices*. Accessed: Apr. 18, 2023. [Online]. Available: https://senso.me/

[43] *Buy Next Generation Full Body Haptic Suit–Bhaptics Tactsuit*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.bhaptics.com/shop

[44] *Haptic Gloves for Virtual Reality and Robotics | Haptx*. Accessed: Apr. 18, 2023. [Online]. Available: https://haptx.com/

[45] *Haptics | Ultraleap*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.ultraleap.com/haptics/

[46] (2021). *Q&A With Lofelt, a Leader in Hd Haptics*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.qualcomm.com/news/onq/2021/05/qa-lofelt-leader-hd-haptics

[47] *Audio to Haptic—How Interhaptics Works?*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.interhaptics.com/tech/how-interhaptics-works/
[48] *Verifying Device Support and User Permission*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/verifying_device_support_and_user_permission
[49] *Arcore Supported Devices | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/devices
[50] *What is a Hologram?—Mixed Reality | Microsoft Learn*. Accessed: Mar. 27, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/discover/hologram
[51] *Microsoft Hololens 2 Vs Microsoft Hololens (comparison)*. Accessed: Mar. 27, 2023. [Online]. Available: https://vr-compare.com/compare?h1=EkSDYv0cW&h2=tSCQxsAA
[52] *Hololens 2*. Accessed: Mar. 27, 2023. [Online]. Available: https://www.microsoft.com/en-us/hololens/hardware
[53] (2021). *Hololens 2 Display Troubleshooting*. Accessed: Mar. 27, 2023. [Online]. Available: https://learn.microsoft.com/en-us/hololens/hololens2-display
[54] (2021). *Hologram Stability—Mixed Reality*. Accessed: Mar. 27, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/hologram-stability
[55] (2021). *Get Your Hololens 2 Ready to Use*. Accessed: Mar. 28, 2023. [Online]. Available: https://learn.microsoft.com/en-us/hololens/hololens2-setup
[56] *Magic Leap 2*. Accessed: Mar. 28, 2023. [Online]. Available: https://www.magicleap.com/magic-leap-2
[57] *Magic Leap 2 Vs Magic Leap 1 (comparison)*. Accessed: Mar. 28, 2023. [Online]. Available: https://vr-compare.com/compare?h1=mt3AEYJu5&h2=1N3k3S4MN
[58] (2023). *Comfort and Content Placement*. Accessed: Mar. 28, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/best-practices/comfort-content-placement
[59] (2023). *Spatial Mapping*. Accessed: Mar. 28, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/features/spatial-mapping
[60] (2023). *Magic Leap 2 Devices*. Accessed: Mar. 28, 2023. [Online]. Available: https://www.magicleap.com/ml2-devices
[61] (2023). *Hand Tracking*. Accessed: Mar. 29, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/api-ref/api/Modules/group_hand_tracking
[62] (2023). *Hand Tracking Overview*. Accessed: Mar. 29, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/unity/input/hand-tracking/unity-hand-tracking-api
[63] *Where To Buy*. Accessed: Mar. 30, 2023. [Online]. Available: https://www.magicleap.com/where-to-buy
[64] (2022). *Magic Leap 2: The Most Immersive Enterprise Ar Device*. Accessed: Mar. 29, 2023. [Online]. Available: https://www.magicleap.com/hubfs/docs/Magic-Leap-2_Product-Spec-Sales-Sheet_09.30.2022_Version-English-4.0.pdf?hsLang=en
[65] *Choosing a Development Environment | Magicleap Developer Documentation*. Accessed: Mar. 30, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/getting-started/developer-environment/
[66] *Openxr Setup | Magicleap Developer Documentation*. Accessed: Mar. 30, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/native/native-openxr-setup/
[67] *Webxr | Magicleap Developer Documentation*. Accessed: Mar. 30, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/features/webxr-viewer/
[68] *Unreal Engine 5 SDK Overview | Magicleap Developer Documentation*. Accessed: Mar. 30, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/unreal/unreal-overview/
[69] *Third-Party Resources | Magicleap Developer Documentation*. Accessed: Mar. 30, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/third-party/third-party-resources/
[70] *Application Simulator | Magicleap Developer Documentation*. Accessed: Mar. 30, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/developer-tools/app-sim/app-sim/
[71] *Ar Cloud | Magicleap Developer Documentation*. Accessed: Mar. 30, 2023. [Online]. Available: https://developer-docs.magicleap.cloud/docs/guides/arcloud/
[72] *Mixed Reality With Passthrough*. Accessed: Apr. 3, 2023. [Online]. Available: https://developer.oculus.com/blog/mixed-reality-with-passthrough/
[73] *Meta Quest Pro Vs Oculus Quest 2 (comparison)*. Accessed: Apr. 3, 2023. [Online]. Available: https://vr-compare.com/compare?h1=-MpSqv-rB&h2=pDTZ02PkT
[74] *Tech Specs*. Accessed: Apr. 3, 2023. [Online]. Available: https://www.meta.com/quest/quest-pro/tech-specs/
[75] *Getting Started With Hand Tracking on Meta Quest Headsets*. Accessed: Apr. 4, 2023. [Online]. Available: https://www.meta.com/help/quest/articles/headsets-and-accessories/controllers-and-hand-tracking/hand-tracking-quest-2/
[76] *Adjust the Fit and Feel of Your Meta Quest Pro*. Accessed: Apr. 4, 2023. [Online]. Available: https://www.meta.com/help/quest/articles/getting-started/getting-started-with-quest-pro/headset-fit/
[77] *Meta Quest Pro Full Light Blocker*. Accessed: Apr. 4, 2023. [Online]. Available: https://www.meta.com/quest/accessories/quest-pro-full-light-blocker/
[78] *Develop With Your Preferred Game Engine*. Accessed: Apr. 5, 2023. [Online]. Available: https://developer.oculus.com/get-started-platform/
[79] *Meta Quest 3: New Mixed Reality Vr Headset—Shop Bundles | Meta Store*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.meta.com/quest/quest-3/
[80] *Compare Quest Vr Headsets: Quest 2 Vs. Quest Pro Vs. Quest 3 | Meta Store*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.meta.com/quest/compare/
[81] *Snapdragon XR2 GEN2 Platform | Qualcomm*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.qualcomm.com/products/mobile/snapdragon/xr-vr-ar/snapdragon-xr2-gen-2-platform
[82] *Zenni Vr Prescription Lenses for Meta Quest 3 | Meta Store*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.meta.com/help/quest/articles/getting-started/getting-started-with-quest-3/zenni-vr-prescription-lenses/
[83] *Start Building With Meta Quest 3*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.oculus.com/blog/start-developing-Meta-Quest-3-tips-performance-mixed-reality/
[84] *Vive XR Elite: Specs*. Accessed: Apr. 5, 2023. [Online]. Available: https://www.vive.com/mea-en/product/vive-xr-elite/specs/
[85] *Vive XR Elite: Overview*. Accessed: Apr. 5, 2023. [Online]. Available: https://www.vive.com/mea-en/product/vive-xr-elite/overview/
[86] *HTC Vive XR Elite*. Accessed: Apr. 5, 2023. [Online]. Available: https://vr-compare.com/headset/htcvivexrelite
[87] *About Vive Wave*. Accessed: Apr. 5, 2023. [Online]. Available: https://hub.vive.com/storage/docs/en-us/AboutViveWave.html
[88] *Vive Wave: Openxr*. Accessed: Apr. 5, 2023. [Online]. Available: https://developer.vive.com/resources/openxr/
[89] *Conformant Products*. Accessed: Apr. 6, 2023. [Online]. Available: https://www.khronos.org/conformance/adopters/conformant-products/openxr
[90] *Varjo XR-3—The Industry's Highest Resolution Mixed Reality Headset | Varjo*. Accessed: Apr. 6, 2023. [Online]. Available: https://varjo.com/products/varjo-xr-3/
[91] *Varjo Launches XR-3 and VR-3 Headsets With Next Generation Hand Tracking*. Accessed: Apr. 6, 2023. [Online]. Available: https://www.ultraleap.com/company/news/press-release/varjo-xr3-vr3-hand-tracking/
[92] *System Requirements for XR-3 and VR-3*. Accessed: Apr. 6, 2023. [Online]. Available: https://varjo.com/use-center/get-started/varjo-headsets/system-requirements/xr-3-vr-3/
[93] *Varjo-Ready Software*. Accessed: Apr. 7, 2023. [Online]. Available: https://varjo.com/varjo-ready-software/
[94] *Varjo Native SDK*. Accessed: Apr. 7, 2023. [Online]. Available: https://developer.varjo.com/docs/native/varjo-native-sdk
[95] *Using Varjo Base*. Accessed: Apr. 7, 2023. [Online]. Available: https://varjo.com/use-center/get-to-know-your-headset/using-varjo-base/
[96] *Varjo Workspace*. Accessed: Apr. 7, 2023. [Online]. Available: https://varjo.com/use-center/get-to-know-your-headset/varjo-workspace/
[97] *Reality Cloud*. Accessed: Apr. 7, 2023. [Online]. Available: https://varjo.com/products/realitycloud/

[98] *Most Advanced XR and VR Software With Varjo Subscriptions for XR-3 and VR-3*. Accessed: Apr. 7, 2023. [Online]. Available: https://varjo.com/products/subscriptions/

[99] *Introducing the XTAL-3—The World's Most Advanced Virtual & Mixed Reality Simulation Headset*. Accessed: Apr. 7, 2023. [Online]. Available: https://vrgineers.com/introducing-the-xtal-3/

[100] *XTAL 3 Mixed Reality*. Accessed: Apr. 7, 2023. [Online]. Available: https://www.xtal.pro/product/xtal-3-mr

[101] *Xtal 3—Mixed Reality | Vrgineers*. Accessed: Apr. 7, 2023. [Online]. Available: https://vrgineers.com/xtal-3-mixed-reality/

[102] *VRG C++ API*. Accessed: Apr. 7, 2023. [Online]. Available: https://portal.vrgineers.com/user-guide/software/vrg-c-api/

[103] *Xtal Goes Wireless—The First Professional Wireless Headset is Here*. Accessed: Apr. 7, 2023. [Online]. Available: https://vrgineers.com/press-release/xtal-goes-wireless-the-first-professional-wireless-headset-is-here/

[104] *Sunset FAQ*. Accessed: Apr. 13, 2023. [Online]. Available: https://lightform.notion.site/Sunset-FAQ-ec66702589a14c12b539bfa10061cba9

[105] *LF2+—lightform*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.com/lf2plus

[106] *LFC—Lightform*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.com/lfc

[107] *Creator—Lightform*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.com/creator

[108] *Audio Reactivity Overview*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.notion.site/Audio-Reactivity-Overview-25fce695173a415f8b1ad3369fdc533c

[109] *OSC Controls*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.notion.site/OSC-Controls-4e455eede2a3406abb3986dcd2e1108c

[110] *Interactive Controllers With Touchosc*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.notion.site/Interactive-Controllers-with-TouchOSC-7f049f5493fb429894c943944d2a9c6b

[111] *Send OSC Messages With Microsoft Kinect*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.notion.site/Send-OSC-Messages-with-Microsoft-Kinect-96104c5cfc184f80a996cfd9190de76e

[112] *Project LFX—Lightform*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.com/lfx

[113] *Design Principles—Project LFX—Lightform*. Accessed: Apr. 18, 2023. [Online]. Available: https://lightform.com/lfx/design-principles

[114] *Madmapper Software*. Accessed: Apr. 18, 2023. [Online]. Available: https://madmapper.com/madmapper/features

[115] *Azure Kinect DK—Develop AI Models | Microsoft Azure*. Accessed: Apr. 18, 2023. [Online]. Available: https://azure.microsoft.com/en-us/products/Kinect-dk

[116] *Is the Hololens 3 Coming Next Year?*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.xrtoday.com/mixed-reality/is-the-hololens-3-coming-next-year/

[117] (2023). *Introducing Apple Vision Pro: Apple's First Spatial Computer—Apple*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/

[118] *Apple Vision Pro—Apple*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.apple.com/apple-vision-pro/

[119] *Holokitx*. Accessed: Apr. 13, 2023. [Online]. Available: https://holokit.io/pages/holokit-x

[120] *Cardboard | Google VR | Google Developers*. Accessed: Apr. 13, 2023. [Online]. Available: https://developers.google.com/vr/discover/cardboard

[121] *Overview—Holokit Docs*. Accessed: Apr. 13, 2023. [Online]. Available: https://docs.holokit.io/for-developers/holokit-unity-sdk/overview

[122] *Holokit: Mixed Reality for Everyone*. Accessed: Apr. 13, 2023. [Online]. Available: https://docubase.mit.edu/tools/holokit/

[123] *Zapbox: Mixed Reality for Everyone By Zappar*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.zappar.com/zapbox/

[124] (2011). *Zappar: World-Leading Augmented Reality Solutions Since*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.zappar.com/

[125] *Zapbox: Mixed Reality for Everyone by Zappar*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.zappar.com/zapbox/unity/

[126] *Monocle*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.brilliantmonocle.com/monocle

[127] *Monocle | Brilliant Documentation*. Accessed: Apr. 13, 2023. [Online]. Available: https://docs.brilliantmonocle.com/monocle/monocle/

[128] *Hands-on: Mojo Vision's Smart Contact Lens is Further Along Than You Might Think*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.roadtovr.com/mojo-vision-smart-contact-lens-ar-hands-on/

[129] *A New Direction for Mojo Vision's Groundbreaking Technology*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.mojo.vision/news/a-new-direction

[130] *The Differences Between 3dof and 6dof, and Why—IEEE Digital Reality*. Accessed: Apr. 18, 2023. [Online]. Available: https://digitalreality.ieee.org/publications/degrees-of-freedom

[131] *Fundamental Concepts | Arcore | Google Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/fundamentals

[132] *Content Placement | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/content/content-placement

[133] *Hit-Test/Hit-Testing-Explainer*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/immersive-web/hit-test/blob/master/hit-testing-explainer.md

[134] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Aug. 2005, pp. 590–596.

[135] *Augmented Reality | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/augmented-reality

[136] *Coordinate Systems—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems

[137] *Depth Adds Realism*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/depth

[138] *Interaction (UX) | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/interaction/ux

[139] Google. (2020). *AR Core Elements*. [Online]. Available: https://play.google.com/store/apps/details/ARCore_Elements?id=com.google.ar.unity.ddelements&gl=CO

[140] *Recording and Playback Introduction*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/recording-and-playback

[141] *Openxr Overview—The Khronos Group Inc*. Accessed: Jul. 26, 2023. [Online]. Available: https://www.khronos.org/openxr/

[142] *Openxr Ecosystem Update*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.khronos.org/assets/uploads/apis/OpenXR-EcoSystem-Updat_Jul20.pdf

[143] *Openxr Conformance Test Suite*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/KhronosGroup/OpenXR-CTS

[144] *Openxr Loader—Design and Operation (With All Registered Extensions)*. Accessed: Apr. 18, 2023. [Online]. Available: https://registry.khronos.org/OpenXR/specs/1.0/loader.html

[145] *Mixed Reality Toolkit 3*. Accessed: Apr. 13, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/

[146] *Immersive Web Developer Home*. Accessed: Apr. 18, 2023. [Online]. Available: https://immersiveweb.dev/

[147] *About Us | W3C*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.w3.org/about/

[148] *Webxr/Explainer*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/immersive-web/webxr/blob/master/explainer.md

[149] *Webxr Device API—Spatial Tracking | Webxr*. Accessed: Apr. 18, 2023. [Online]. Available: https://immersive-web.github.io/webxr/spatial-tracking-explainer.html

[150] *Webxr Plane Detection Module*. Accessed: Apr. 18, 2023. [Online]. Available: https://immersive-web.github.io/real-world-geometry/plane-detection.html

[151] *Webxr Marker Tracking Module*. Accessed: Apr. 18, 2023. [Online]. Available: https://immersive-web.github.io/marker-tracking/

[152] *Arkit 6—Augmented Reality—Apple Developer*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/augmented-reality/arkit/

[153] *Arconfiguration.Scenereconstruction | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arconfiguration/scenereconstruction

[154] *Islightestimationenabled | Apple Developer Documentation*. Accessed: Apr. 13, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arconfiguration/2923546-islightestimationenabled
[155] *Environmenttexturing | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arworldtrackingconfiguration/2977509-environmenttexturing
[156] *Framesemantics | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arconfiguration/3089121-framesemantics
[157] *Arbodytrackingconfiguration | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arbodytrackingconfiguration
[158] *Arkit in Visionos | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arkit_in_visionos
[159] *Arfacetrackingconfiguration | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arfacetrackingconfiguration
[160] *Arworldtrackingconfiguration | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arworldtrackingconfiguration
[161] *Content Anchors | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/content_anchors
[162] *Arobjectscanningconfiguration | Apple Developer Documentation*. Accessed: Apr. 13, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arobjectscanningconfiguration
[163] *Argeotrackingconfiguration | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/argeotrackingconfiguration
[164] *Configuration Objects*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/configuration_objects
[165] *Choosing Which Camera Feed To Augment*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/choosing_which_camera_feed_to_augment
[166] *Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/appclips/interacting_with_appclip_codes_in_ar
[167] *Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/appclips/creating_appclip_codes
[168] *Quick Look Gallery—Augmented Reality—Apple Developer*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/augmented-reality/quick-look/
[169] *Realitykit Overview—Augmented Reality—Apple Developer*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/augmented-reality/realitykit/
[170] *Creating 3D Content With Reality Composer | Apple Developer Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/realitykit/creating-3d-content-with-reality-composer
[171] *Roomplan Overview—Augmented Reality—Apple Developer*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/augmented-reality/roomplan/
[172] *Introducing Reality Converter—Latest News—-Aapple Developer*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/news/?id=01132020a
[173] *Build New Augmented Reality Experiences That Seamlessly Blend the Digital and Physical Worlds | Arcore | Google Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar
[174] *Place Objects Instantly*. Accessed: Apr. 13, 2023. [Online]. Available: https://developers.google.com/ar/develop/instant-placement
[175] *Get the Lighting Right*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/lighting-estimation
[176] *Smooth Camera Preview With Electronic Image Stabilization (ESI) | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/electronic-image-stabilization
[177] *Add Dimension to Images*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/augmented-images
[178] *Augmented Faces Introduction*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/augmented-faces
[179] *Cloud Anchors Allow Different Users to Share Ar Experiences*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/cloud-anchors
[180] *Arcore Cloud Anchor API Deprecation Policy | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/cloud-anchors/cloud-anchor-deprecation-policy
[181] *Build Global-Scale, Immersive, Location-based AR Experiences With the Arcore Geospatial API*. Accessed: Apr. 13, 2023. [Online]. Available: https://developers.google.com/ar/develop/geospatial
[182] *Use Buildings and Terrain Around You on Android Sdk (kotlin/java) | Arcore | Google for Developers*. Accessed: Apr. 13, 2023. [Online]. Available: https://developers.google.com/ar/develop/java/geospatial/streetscape-geometry
[183] *Geospatial Creator | Arcore | Google for Developers*. Accessed: Apr. 13, 2023. [Online]. Available: https://developers.google.com/ar/geospatialcreator/intro
[184] *Machine Learning With Arcore*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/machine-learning
[185] *Understand the User's Environment With the Scene Semantics API | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/develop/scene-semantics
[186] *Vuforia Enterprise Augmented Reality (AR) Software | PTC*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.ptc.com/en/products/vuforia
[187] *Pricing and Licensing Options | Vuforia Library*. Accessed: Apr. 13, 2023. [Online]. Available: https://library.vuforia.com/faqs/pricing-and-licensing-options
[188] *Vuforia Engine Overview | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/getting-started/vuforia-features
[189] *Vumarks | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.vuforia.com/library/objects/vumarks
[190] *Vumark Design Guide | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/vumarks/vumark-design-guide
[191] *Barcode Scanner | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/objects/barcode-scanner
[192] *Area Targets | Vuforia Library*. Accessed: Apr. 13, 2023. [Online]. Available: https://library.vuforia.com/environments/area-targets
[193] *External Camera Support | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/platform-support/external-camera
[194] *Cloud Recognition | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/objects/cloud-recognition
[195] *Vuforia Fusion | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/environments/vuforia-fusion
[196] *Recommended Devices | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/platform-support/recommended-devices
[197] *Supported Versions | Vuforia Library*. Accessed: Apr. 18, 2023. [Online]. Available: https://library.vuforia.com/platform-support/supported-versions
[198] *Vuforia Expert Capture: AR Knowledge Capture Tools | PTC*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.ptc.com/en/products/vuforia/vuforia-expert-capture
[199] *Vuforia Studio Augmented Reality for Industrial Enterprise | PTC*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.ptc.com/en/products/vuforia/vuforia-studio
[200] *Vuforia Chalk Augmented Reality (AR) Remote Assistance | PTC*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.ptc.com/en/products/vuforia/vuforia-chalk
[201] *Introducing MRTK for Unreal*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unreal/unreal-mrtk-introduction
[202] (2022). *What is Mixed Reality Toolkit 2*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05
[203] *What are the UX Tools?*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/microsoft/MixedReality-UXTools-Unreal/blob/public/0.12.x-UE5.0/README.md

[204] *What is Graphics Tools?*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/microsoft/MixedReality-GraphicsTools-Unreal/blob/main/README.md

[205] (2022). *Input Overview—MRTK 2 | Microsoft Learn*. Accessed: Apr. 13, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/overview?view=mrtkunity-2022-05.

[206] (2022). *Extension Services—MRTK2 | Microsoft Learn*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/extensions/extension-services?view=mrtkunity-2022-05

[207] (2022). *Camera System Overview—MRTK 2 | Microsoft Learn*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/camera-system/camera-system-overview?view=mrtkunity-2022-05

[208] (2022). *Profiles—MRTK 2 | Microsoft Learn*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/profiles/profiles?view=mrtkunity-2022-05

[209] *Input Simulation*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/microsoft/MixedReality-UXTools-Unreal/blob/public/0.12.x-UE5.0/Docs/InputSimulation.md

[210] (2022). *Input Simulation Service—MRTK 2 | Microsoft Learn*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input-simulation/input-simulation-service?view=mrtkunity-2022-05

[211] *Input Simulation—MRTK3 | Microsoft Learn*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-input/packages/input/input-simulation

[212] *Wikitude Augmented Reality: The World's Leading Cross-platform AR SDK*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.wikitude.com/

[213] *What's the Difference Between the Professional and the Expert Editions? : Wikitude*. Accessed: Apr. 18, 2023. [Online]. Available: https://support.wikitude.com/support/solutions/articles/5000854979-what-s-the-difference-between-the-professional-and-the-expert-editions-

[214] *Augmented Reality—Wikitude SDK Full Features Overview*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.*Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/app

[215] *Object and Scene Recognition*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.wikitude.*Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/app

[216] *Instant Tracking*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.wikitude.*Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/app_clips/interacting_

[217] *Release Notes Wikitude Sdk Professional Edition*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.wikitude.*Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/app

[218] *Universal Render Pipeline Overview | Universal RP | 16.0.3*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unity3d.*Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/app

[219] *Advanced Rendering | Wikitude Documentation—Expert Edition*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.wikitude.*Interacting With App Clip Codes in AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.apple.com/documentation/app

[220] *Wikitude SDK Studio User Manual*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.wikitude.com/external/doc/documentation/studio/introduction.html

[221] *Solutions: Wikitude*. Accessed: Apr. 18, 2023. [Online]. Available: https://support.wikitude.com/support/solutions

[222] *Download Wikitude 3D Encoder for Windows—Wikitude*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.wikitude.com/download-wikitude-3d-encoder-for-windows/

[223] *Wikitude Store: Find Best Pricing for Your Augmented Reality Experiences*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.wikitude.com/store/

[224] *Snapdragon Spaces Augmented Reality SDK | Snapdragon Spaces*. Accessed: Apr. 18, 2023. [Online]. Available: https://spaces.qualcomm.com/sdk/

[225] *General Features | Snapdragon Spaces*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.spaces.qualcomm.com/common/features/GeneralFeatures.html

[226] *Environmental Understanding | Snapdragon Spaces*. Accessed: Apr. 13, 2023. [Online]. Available: https://docs.spaces.qualcomm.com/common/features/EnvironmentalFeatures.html

[227] *Interaction | Snapdragon Spaces*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.spaces.qualcomm.com/common/features/Interaction.html

[228] *Openxr for Snapdragon Spaces | Snapdragon Spaces*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.spaces.qualcomm.com/common/architecture/OpenXRForSpaces.html

[229] *Ar-media Features—Inglobe Technologies*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.inglobetechnologies.com/ar-media/features/

[230] *Introduction—Inglobe Technologies*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.inglobetechnologies.com/ar-media/documentation/overview-introduction/

[231] *Ar-media Pricing—Inglobe Technologies*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.inglobetechnologies.com/ar-media/pricing/

[232] *Artoolkit Home Page*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.hitl.washington.edu/artoolkit.html

[233] *Home*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/artoolkitx/artoolkitx/wiki

[234] *Creating and Using Multi Square Marker Sets*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/artoolkitx/artoolkitx/wiki/Creating-and-using-multi-square-marker-sets

[235] *Artoolkit Documentation (History)*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.hitl.washington.edu/artoolkit/documentation/history.htm

[236] *About Us—Artoolworks*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.artoolworks.com/corporate/about-us.html

[237] *Artoolkitx*. Accessed: Apr. 18, 2023. [Online]. Available: http://www.artoolkitx.org/

[238] *Read Me for Artoolkitx*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/artoolkitx/artoolkitx

[239] *Welcome to Artoolkitx for Unity*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/artoolkitx/arunityx

[240] *Github—Artoolkitx/jsartoolkit5: Javascript Artoolkit V5.x*. Accessed: Apr. 13, 2023. [Online]. Available: https://github.com/artoolkitx/jsartoolkit5

[241] *Products—Artoolworks*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.artoolworks.com/products.html

[242] *Licensing—Artoolworks*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.artoolworks.com/products/licensing.html

[243] *Niantic Lightship*. Accessed: Jan. 8, 2024. [Online]. Available: https://lightship.dev/

[244] *Meta Spark Studio—Create Immersive AR Experiences*. Accessed: Jan. 8, 2024. [Online]. Available: https://spark.meta.com/

[245] *Tiktok Effect House*. Accessed: Jan. 8, 2024. [Online]. Available: https://effecthouse.tiktok.com/

[246] *Real-Time 3D Development Platform & Editor| Unity*. Accessed: Apr. 18, 2023. [Online]. Available: https://unity.com/products/unity-engine

[247] (2023). *Unity—-Manual: Creating and Using Scripts*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unity3d.com/2023.2/Documentation/Manual/CreatingAndUsingScripts.html

[248] *Augmented Reality Game Design Software for Apps | Unity*. Accessed: Apr. 18, 2023. [Online]. Available: https://unity.com/unity/features/ar

[249] *Unity's Ar Foundation Framework | Cross Platform Augmented Reality Development Software | Unity*. Accessed: Apr. 13, 2023. [Online]. Available: https://unity.com/unity/features/arfoundation

[250] *Advanced Workflows for AR Developers | Unity Mars*. Accessed: Apr. 18, 2023. [Online]. Available: https://unity.com/products/unity-mars

[251] *The AR Companion App is Now Available | Unity Blog*. Accessed: Apr. 18, 2023. [Online]. Available: https://blog.unity.com/engine-platform/the-ar-companion-app-is-now-available
[252] *Xr Interaction Toolkit | Xr Interaction Toolkit | 2.4.3*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit
[253] *Unity As a Library for Native Mobile Apps Written in Objective C & Java | Ar for Android & Ios | Unity*. Accessed: Apr. 18, 2023. [Online]. Available: https://unity.com/features/unity-as-a-library
[254] *Plans and Pricing*. Accessed: Apr. 18, 2023. [Online]. Available: https://unity.com/pricing
[255] *The Most Powerful Real-time 3D Creation Tool—0unreal Engine*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.unrealengine.com/en-U.S
[256] *Unreal Engine Programming and Scripting | Unreal Engine 5.2 Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unrealengine.com/5.2/en-U.S./unreal-engine-programming-and-scripting/
[257] *Supported XR Devices in Unreal Engine | Unreal Engine 5.2 Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unrealengine.com/5.2/en-U.S./supported-xr-devices-in-unreal-engine/
[258] *Developing for Handheld Augmented Reality Experiences in Unreal Engine | Unreal Engine 5.2 Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unrealengine.com/5.2/en-U.S./developing-for-handheld-augmented-reality-experiences-in-unreal-engine/
[259] *Developing for Head-mounted Experiences With Openxr in Unreal Engine | Unreal Engine 5.2 Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unrealengine.com/5.2/en-U.S./developing-for-head-mounted-experiences-with-openxr-in-unreal-engine/
[260] *Magic Leap Development | Unreal Engine 4.27 Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unrealengine.com/4.27/en-U.S./SharingAndReleasing/XRDevelopment/AR/ARPlatforms/MagicLeap/
[261] *Sharing XR Experiences in Unreal Engine | Unreal Engine 5.2 Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unrealengine.com/5.2/en-U.S./sharing-xr-experiences-in-unreal-engine/
[262] *Realityscan | Free to Download 3D Scanning App for Mobile—Unreal Engine*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.unrealengine.com/en-U.S./realityscan
[263] *Recording Face Animation on Ios Device in Unreal Engine | Unreal Engine 5.2 Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.unrealengine.com/5.2/en-U.S./recording-face-animation-on-ios-device-in-unreal-engine/
[264] *Metahuman | Realistic Person Creator—Unreal Engine*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.unrealengine.com/en-U.S./metahuman
[265] *Godot Engine—Free and Open Source 2D and 3Dd Game Engine*. Accessed: Apr. 18, 2023. [Online]. Available: https://godotengine.org/
[266] *Scripting Languages—Godot Engine (Stable) Documentation in English*. Accessed: Apr. 18, 2023. [Online]. Available: https://docs.godotengine.org/en/stable/getting_started/step_by_step/scripting_languages.html
[267] *Godot 4.0 Sets Sail: All Aboard for New Horizons*. Accessed: Apr. 13, 2023. [Online]. Available: https://godotengine.org/article/godot-4-0-sets-sail/
[268] *Godot 3.2 ARVR Update*. Accessed: Apr. 18, 2023. [Online]. Available: https://godotengine.org/article/godot-3-2-arvr-update/
[269] *Download for Windows—Godot Engine*. Accessed: Apr. 13, 2023. [Online]. Available: https://godotengine.org/download/windows/
[270] *Introduction to the Dom—Web APIS | Mdn*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.mozilla.org/en-U.S./docs/Web/API/Document_Object_Model/Introduction
[271] *Dom-Overlays/Explainer.Md At Master*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/immersive-web/dom-overlays/blob/master/explainer.md
[272] *Model-Viewer*. Accessed: Apr. 18, 2023. [Online]. Available: https://modelviewer.dev/
[273] *Model-Viewer Faq*. Accessed: Apr. 18, 2023. [Online]. Available: https://modelviewer.dev/docs/faq.html.
[274] *Model-Viewer Postprocessing Examples*. Accessed: Apr. 18, 2023. [Online]. Available: https://modelviewer.dev/examples/postprocessing/
[275] *Model Editor*. Accessed: Apr. 18, 2023. [Online]. Available: https://modelviewer.dev/editor/
[276] *ARJS Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://ar-js-org.github.io/AR.js-Docs/
[277] *Marker Based—AR.JS Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://ar-js-org.github.io/AR.js-Docs/marker-based/
[278] *Ui and Events—Ar.js Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://ar-js-org.github.io/AR.js-Docs/ui-events/
[279] *Introduction—A-Frame*. Accessed: Apr. 18, 2023. [Online]. Available: https://aframe.io/docs/1.4.0/introduction/
[280] *Entity-component-system—A-Frame*. Accessed: Apr. 18, 2023. [Online]. Available: https://aframe.io/docs/1.4.0/introduction/entity-component-system.html
[281] *Experiment With Ar and A-frame—A-Frame*. Accessed: Apr. 18, 2023. [Online]. Available: https://aframe.io/blog/webxr-ar-module/
[282] *Creating Augmented Reality With Ar.Js and A-frame—A-Frame*. Accessed: Apr. 13, 2023. [Online]. Available: https://aframe.io/blog/arjs/
[283] *Webxr-A-Frame*. Accessed: Apr. 18, 2023. [Online]. Available: https://aframe.io/docs/1.4.0/components/webxr.html
[284] *Ar-Hit-Test-A-frame*. Accessed: Apr. 18, 2023. [Online]. Available: https://aframe.io/docs/1.4.0/components/ar-hit-test.html
[285] *Reflection—A-Frame*. Accessed: Apr. 18, 2023. [Online]. Available: https://aframe.io/docs/1.4.0/components/reflection.html
[286] *Babylon.Js: Powerful, Beautiful, Simple, Open—Wb-based 3D At Its Best*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.babylonjs.com/
[287] *Babylon.JS Specifications*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.babylonjs.com/specifications/
[288] *Webxr Augmented Reality Features | Babylon.js Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://doc.babylonjs.com/features/featuresDeepDive/webXR/webXRARFeatures.
[289] *Mrtk 2.x for Babylon.js | Babylon.js Documentation*. Accessed: Apr. 13, 2023. [Online]. Available: https://doc.babylonjs.com/features/featuresDeepDive/gui/mrtk
[290] *Babylon.js Sandbox—View GLTF, GLB, OBJ and Babylon Files*. Accessed: Apr. 18, 2023. [Online]. Available: https://sandbox.babylonjs.com/
[291] *Playground | Babylon.Js Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://doc.babylonjs.com/toolsAndResources/thePlayground
[292] *The Node Material Editor | Babylon.Js Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://doc.babylonjs.com/toolsAndResources/nme
[293] *The Gui Editor | Babylon.Js Documentation*. Accessed: Apr. 18, 2023. [Online]. Available: https://doc.babylonjs.com/toolsAndResources/guiEditor
[294] *8th Wall | Product—Webar SDK for World Tracking, Face Effects, Image Targets, Inline AR*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.8thwall.com/products-web
[295] *Introduction | 8th Wall*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.8thwall.com/docs/
[296] *Changelog | 8th Wall*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.8thwall.com/docs/changelog/
[297] *Shared AR Module | 8th Wall | 8th Wall*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.8thwall.com/8thwall/modules/shared-ar
[298] *Qr 8CODE | 8th Wall*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.8thwall.com/docs/guides/projects/qr-8code/
[299] *Requirements | 8th Wall*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.8thwall.com/docs/getting-started/requirements/
[300] *8th Wall | Open Source Licenses*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.8thwall.com/open-source-licenses
[301] *Introducing the New 8th Wall Project Library Featuring Over 30+ Projects | 8th Wall*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.8thwall.com/blog/post/41172589178/introducing-the-new-8th-wall-project-library-featuring-over-30-projects
[302] *Needle Tools*. Accessed: Apr. 13, 2023. [Online]. Available: https://needle.tools/
[303] *Virtual and Augmented Reality | Needle Engine Documentation*. Accessed: Apr. 13, 2023. [Online]. Available: https://engine.needle.tools/docs/xr.html

[304] *Playcanvas Webgl Game Engine*. Accessed: Apr. 13, 2023. [Online]. Available: https://playcanvas.com/

[305] *Ar | Learn Playcanvas*. Accessed: Apr. 13, 2023. [Online]. Available: https://developer.playcanvas.com/en/user-manual/xr/ar/

[306] *Xrestimatedlight—Three.js Docs*. Accessed: Apr. 13, 2023. [Online]. Available: https://threejs.org/docs/index.html

[307] *Libraries and Plugins—Three.js Docs*. Accessed: Apr. 13, 2023. [Online]. Available: https://threejs.org/docs/index.html

[308] *General Info*. Accessed: Apr. 13, 2023. [Online]. Available: https://p5xr.org/

[309] *Home | P5.js*. Accessed: Apr. 13, 2023. [Online]. Available: https://p5js.org/

[310] *Device and Browser Support*. Accessed: Apr. 13, 2023. [Online]. Available: https://p5xr.org/

[311] *Github—Pmndrs/react-xr: Vr/ar With React-three-fiber*. Accessed: Apr. 13, 2023. [Online]. Available: https://github.com/pmndrs/react-xr

[312] *Github—Pmndrs/react-three-fiber: A React Renderer for Three.js*. Accessed: Apr. 13, 2023. [Online]. Available: https://github.com/pmndrs/react-three-fiber

[313] *Verge3d: An Artist-friendly Toolkit for 3D Web Experiences—Soft8soft*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.soft8soft.com/verge3d/

[314] *Ar/Vr Development—Soft8soft*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.soft8soft.com/docs/manual/en/introduction/AR-VR-development.html

[315] *Verge3d Licensing Options–Soft8soft*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.soft8soft.com/licensing/

[316] *Verge3d Ultimate Web Interactive Suite—Soft8soft*. Accessed: Apr. 13, 2023. [Online]. Available: https://www.soft8soft.com/product/verge3d-ultimate-web-interactive-suite/

[317] *Zapworks: The Most Powerful All-in-One Webar Platform*. Accessed: Apr. 13, 2023. [Online]. Available: https://zap.works/

[318] *Zapworks Webar: Publish AR Experiences Directly to the Web*. Accessed: Apr. 13, 2023. [Online]. Available: https://zap.works/webar/

[319] *Zapworks Webxr: Rich and Immersive XR Experiences*. Accessed: Apr. 13, 2023. [Online]. Available: https://zap.works/webxr/

[320] *Zapworks Pricing: Choose the Right Plan for You*. Accessed: Apr. 13, 2023. [Online]. Available: https://zap.works/pricing/

[321] *What is Wonderland Engine | Wonderland Engine*. Accessed: Apr. 13, 2023. [Online]. Available: https://wonderlandengine.com/about/what-is-wle/

[322] *Quick Start—Augmented Reality | Wonderland Engine*. Accessed: Apr. 13, 2023. [Online]. Available: https://wonderlandengine.com/getting-started/quick-start-ar/

[323] *Meet Our Products | Wonderland Engine*. Accessed: Apr. 13, 2023. [Online]. Available: https://wonderlandengine.com/pricing/

[324] *Mastering Apple's Ar Guidelines: An Exploration Into Designing for AR | By Martina Sartor | Ux Planet*. Accessed: Apr. 18, 2023. [Online]. Available: https://uxplanet.org/mastering-apples-ar-guidelines-an-exploration-into-designing-for-ar-a8632d84ba6e

[325] *Brainstorming At the D.School—YouTube*. Accessed: Jan. 12, 2024. [Online]. Available: https://www.youtube.com/watch?v=cmoWCSyujPY

[326] *Environment | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/environment/definition

[327] *Experience Size | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/environment/experience-size

[328] *Holographic Frame—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/holographic-frame

[329] *Scale—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/scale

[330] *Content Manipulation | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/content/content-manipulation

[331] *Types of Mixed Reality Apps—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/discover/types-of-mixed-reality-apps

[332] *Design Process for Mixed Reality—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/discover/case-study-expanding-the-design-process-for-mixed-reality

[333] *7 Tips to Consider When Designing for Hololens | By Jan Marek | Inloopx | Medium*. Accessed: Apr. 18, 2023. [Online]. Available: https://medium.com/inloopx/7-tips-to-consider-when-designing-for-hololens-67d7b09d3e86

[334] *A Quick Guide To Designing for Augmented Reality on Mobile (Part 2) | By Bushra Mahmood | Medium*. Accessed: Apr. 18, 2023. [Online]. Available: https://medium.com/

[335] *Color, Light, and Materials—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/color-light-and-materials

[336] *Designing Content for Holographic Display—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/designing-content-for-holographic-display

[337] *Gaze and Commit—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/gaze-and-commit

[338] *Comfort—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/comfort

[339] *A Quick Guide To Designing for Augmented Reality on Mobile (Part 4) | By Bushra Mahmood | Medium*. Accessed: Apr. 18, 2023. [Online]. Available: https://medium.com/

[340] *Mixed Reality Ux Elements—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/app-patterns-landingpage

[341] *Spatial Mapping—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping

[342] *Augmented Reality Design Guidelines | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design

[343] *Realism | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/content/realism

[344] *Hololens Environment Considerations | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/hololens/hololens-environment-considerations

[345] *A Quick Guide to Designing for Augmented Reality on Mobile (Part 1) | By Bushra Mahmood | Medium*. Accessed: Apr. 18, 2023. [Online]. Available: https://medium.com/

[346] *Instinctual Interactions—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/interaction-fundamentals

[347] *UI Elements | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/interaction/ui

[348] *A Quick Guide To Designing for Augmented Reality on Mobile (Part 3) | By Bushra Mahmood | Medium*. Accessed: Apr. 18, 2023. [Online]. Available: https://medium.com/

[349] *Typography—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/typography

[350] *Voice Input—Mixed Reality | Microsoft Learn*. Accessed: Aug. 4, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/voice-input

[351] *Movement | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/user/movement

[352] *Safety and Comfort | Arcore | Google for Developers*. Accessed: Apr. 18, 2023. [Online]. Available: https://developers.google.com/ar/design/user/safety-comfort

[353] *Oculus Connect 4 | Day 2 Keynote: Carmack Unscripted—YouTube*. Accessed: Jan. 12, 2024. [Online]. Available: https://www.youtube.com/watch?v=vlYL16-NaOw

[354] *Google Glass Enterprise Edition 2: Full Specification—Vrcompare*. Accessed: Apr. 7, 2023. [Online]. Available: https://vr-compare.com/headset/googleglassenterpriseedition2

[355] *Epson Moverio Bt-40s: Full Specification - Vrcompare*. Accessed: Apr. 7, 2023. [Online]. Available: https://vr-compare.com/headset/epsonmoveriobt-40s
[356] *Nreal Light: Full Specification—vrcompare*. Accessed: Apr. 7, 2023. [Online]. Available: https://vr-compare.com/headset/nreallight
[357] *Nreal Air: Full Specification—Vrcompare*. Accessed: Apr. 7, 2023. [Online]. Available: https://vr-compare.com/headset/nrealair
[358] *Vuzix M400: Full Specification—Vrcompare*. Accessed: Apr. 7, 2023. [Online]. Available: https://vr-compare.com/headset/vuzixm400

**FAYAZ MOHAMED HANEEFA** received the B.Sc. degree in computer engineering from Khalifa University, Abu Dhabi, United Arab Emirates, in 2022. He is currently a Research Assistant with the Center for Cyber-Physical Systems, Khalifa University. His research interests include XR, computer graphics, and computer animation.

**ABDULHADI SHOUFAN** (Member, IEEE) received the Dr.-Ing. degree from Technische Universität Darmstadt, Germany, in 2007. He is currently an Associate Professor of computer and information engineering with Khalifa University, Abu Dhabi. His research interests include drone security, safe operation, embedded security, cryptography hardware, learning analytics, and engineering education.

**ERNESTO DAMIANI** (Senior Member, IEEE) is currently a Full Professor with the UniversitÃ degli Studi di Milano, Italy, the Senior Director of the Robotics and Intelligent Systems Institute, and the Director of the Center for Cyber-Physical Systems (C2PS), Khalifa University, United Arab Emirates. He is also the Leader of the Big Data Area, Etisalat British Telecom Innovation Center (EBTIC), and the President of the Consortium of Italian Computer Science Universities (CINI). He is also a part of the ENISA Ad-Hoc Working Group on Artificial Intelligence Cybersecurity. He has pioneered model-driven data analytics. He has authored more than 650 Scopus-indexed publications and several patents. His research interests include cyber-physical systems, big data analytics, edge/cloud security and performance, artificial intelligence, and machine learning. He was a recipient of the Research and Innovation Award from the IEEE Technical Committee on Homeland Security, the Stephen Yau Award from the Service Society, the Outstanding Contributions Award from IFIP TC2, the Chester-Sall Award from IEEE IES, the IEEE TCHS Research and Innovation Award, and the Doctorate Honoris Causa from INSA-Lyon, France, for his contribution to big data teaching and research.