GENERIC AND ACCESSIBLE GESTURE CONTROLLED AUGMENTED REALITY PLATFORM

Arya Rajiv Chaloli, K Anjali Kamath,
Divya T Puranam and Prof. Preet Kanwal

Department of Computer Science, PES University Bangalore, India

ABSTRACT

Augmented Reality (AR) is one of the most popular trends in, technology today. Its accessibility has heightened as new smartphones and other devices equipped with depth-sensing cameras and other AR- related technologies are being introduced into the market. AR helps the user view the real-world environment with virtual objects augmented in it. With advances in AR technology, Augmented Reality is being used in a variety of applications, from medicine to games like Pokémon Go, and to retail and shopping applications that allow us to try on clothes and accessories from the comfort of our homes. In the current times, being hands-off is utterly important due to the widespread attack of the COVID19 pandemic. Thus, an application where the necessity to touch a screen is removed would be highly relevant. In such a scenario, AR comes into play. Essentially, it helps to convert the contents on a physical screen to a virtual object that is seamlessly augmented into reality and the user interacts only with the virtual object, thus avoiding any requirement to touch the actual physical screen and making the whole system touch-free.

Hence, the paper aims to propose an Augmented Reality application system that can be integrated into our day-to-day life. With the advent of technology and the digitization of almost all interfaces around us, the opportunity of augmenting these digitized resources has escalated. To demonstrate a generic application that can be used along with any digitized resource, an AR system will be implemented in the project, that can control a personal computer (PC) remotely through hand gestures made by the user on the augmented interface. The methodology proposed targets to build a system that is both generic and accessible. The application is made generic by making the AR system be able to provide an AR interface to any application that can run on a regular PC. The accessibility of the AR system is improved by making it compatible to work on any normal smartphone with a regular camera. There is no necessity for a depth- sensing camera, which is a requirement of popular AR toolkits like ARCore and ARKit.

KEYWORDS

Augmented reality, Gesture Recognition, Generic, Application, Technology, Accessibility.

1. INTRODUCTION

Augmented Reality (AR) provides an interactive experience of virtual objects augmented into the real-world environment by enhancing both real and virtual objects using computer-generated perceptual information, which may span across multiple sensory modalities. It can also be defined as a system consisting of 3 parts namely, the combination of real and virtual worlds, real-time interaction, and the accurate recognition and registration of real and virtual objects. Using this, objects appear as though they are part of our environment but in reality, it is just a simulation. In

David C. Wyld et al. (Eds): CST, NLMLT, DMS, CLBD, ITCS, VLSIE - 2022
pp. 01-10, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121901

this way, AR can be used to alter one's ongoing perception of the real world and create new experiences [1, 2].

Modified reality systems are of three types. Augmented Reality adds virtual objects to a live view of the real world by making use of the camera on a smartphone or a headset. Virtual Reality, on the other hand, is a completely virtual experience that contains no aspects of the real world. This is done using VR devices like HTC Vive, Oculus Rift, or Google Cardboard. A Mixed Reality (MR) experience combines the elements of both AR and VR, i.e., it is the interaction between real-world and virtual objects.

Augmented Reality is a newly emerging field and has been used in fields such as medical training, retail, design and modelling, education [3], and entertainment, in the past few years. However, due to the field still being in its formative stages, it still lacks the genericness that it should have and is rather very specific to the applications that it is being used for. Some of the current limitations [4] in the field of Augmented Reality include:

- Hardware:
  - Equipment: Very few mobile devices available in the market are equipped with room mapping or depth sensing technology.
  - Specifications: Limited processing power, a small amount of memory, and limited storage available during the deployment phase (phone apps).
- Blending with Reality: Rendering digital data into meaningful and easily understandable graphics and scaling it to fit the perspective of the visual field is challenging.
- AR education: Most consumers are not familiar with AR technology and do not see its applications in their daily lives.
- Possibility of physical harm: There have been many cases of people hurting themselves while playing the popular AR game Pokémon Go.
- Lack of proven business models: There are not many industries that have found an AR-related business model that will work long term, except the gaming industry.
- Privacy and Security: Augmented Reality identification and security policies are not quite developed.

The proposed methodology aims to develop a generic application with high accessibility that would reduce contact with physical surfaces. The user would operate through an AR screen instead of using the surface directly. However, for demonstration and feasibility testing, the scope is restricted to the deployment of an AR interface for a browser application (that runs on a remote PC) on android smartphones With the COVID19 pandemic on the rise, there is a heightened need to maintain social distancing. Washing and sanitizing are of the utmost importance when anything in a public place is touched. As it goes, prevention is better than cure, and hence such situations must be avoided. Through the project, the user can just look at the screen through his/her phone and perform operations using gestures [5, 6] in the air and his/her interaction with the software is completely handsfree. Hence one can avoid coming in contact with public surfaces. Additionally, the screen is completely restricted to the user and hence can provide enhanced security and privacy.

2. RELEVANT WORK

Broadly, the relevant work in this domain falls under three categories: gesture recognition, augmented reality, and gesture recognition in augmented reality.

2.1. Gesture Recognition

There are a variety of methods that are used for extracting the necessary information that is required for gesture recognition systems which include hardware devices like data gloves and colour markers and using the appearance of the hand and skin colour to segment the hand.

Authors Khan, Rafiqul Zaman, and Noor Adnan Ibraheem in [7] explains how gesture recognition systems are mainly classified into three steps: extraction method, features extraction, and classification.

- The extraction method/segmentation is the first step. It consists of dividing the input image into regions separated by boundaries. This is done differently based on whether the gesture is static or dynamic. Some of the methods used to model the hand are the Gaussian Model, Gaussian Mixture Model, and histogram-based techniques.
- In the second step, the features of the segmented image are extracted. Some extraction methods use the contour and silhouette of the hand while others use fingertip positions, palm centres, etc.
- The last step is gesture classification. Various techniques and models like neural networks, FSMs, PCA, Fuzzy CMeans clustering, and genetic algorithms are used for this purpose.

The authors of [7] compare the recognition methods used in hand gesture recognition systems. A summary of extraction methods, feature representation, and recognition of hand gesture recognition systems is listed. They also describe the various methods used for gesture recognition like Neural Networks, HMMs, fuzzy, C-means clustering, etc (which are alternatives for orientation histogram for feature representation). Reference [7] provides more clarity and an insight into the various applications, internal working, limitations, and other specifications of different gesture recognition systems.

2.2. Augmented Reality

Reference [8] deals with the fundamentals of Augmented Reality. As described in [8], the four fundamental parts of Augmented Reality are AR components, AR devices, the applications of Augmented Reality, and visualization issues that could be encountered.

The scene generator, tracking system, and display are the components of an Augmented Reality system. The scene generator is used to set the scene of the Augmented Reality environment. This is a challenge since, unlike a Virtual Reality system, an Augmented Reality system needs to account for the real environment that the virtual object is being placed in. The second component is the tracking system used to track both the virtual and real objects in the Augmented Reality environment. The last component is the display features of the Augmented Reality system.

From [8], the complexities of an AR System are understood, and the importance of a good interface is highlighted multiple times. However, little or no information regarding gesture recognition in Augmented Reality systems is dealt with in the paper. And all the extra information like, types of devices that could be used for AR systems, etc, though helpful for understanding the domain, does not directly align with the project.

2.3. Gesture Recognition in Augmented Reality

Different gesture recognition techniques are proposed by [9] for three basic interaction categories (translation, rotation, and scaling) in a Leap Motion Controller, Augmented Reality framework. In [9], the proposed model is implemented using the Leap Motion Controller (LMC) along with the Unity3D platform. The Leap Motion Controller is a camera sensor developed by Leap Motion that senses natural hand movements and the position of our fingers and allows us to interact with the system by using gestures like swiping, grabbing, pinching, and rotating. In the authors' prototype, they design two different gestures which are of the high and low level of naturalism for three interaction categories, which are translation, scaling, and rotation.

Reading [9] gave us an overview of the types of gestures used in hand gesture recognition systems like translation, scaling, and rotation and helped decide which gestures need to be supported in the project. The user's expectations from an AR application and what could be done to make the user's performance and experience better were also observed from [9].

In [10], the authors aim to develop a natural interaction technique that allows the manipulation of virtual objects on handheld augmented reality devices in 3D space. This method relies on the identification of the position and the movements of the user's fingers. The recognized gestures are then mapped to the corresponding manipulations of the virtual objects in the AR scene. The method implemented in [10] provides 6 degrees of freedom manipulations using natural finger-based gestures for rotating, translating, or scaling a virtual object in a handheld AR system. Markerless 3D gesture-based interaction design has two major components to it: Object Selection and Canonical Manipulations. Reference [10] provided an insight into how the user's hand could be segmented from the surroundings. It also highlighted the different ways in which virtual objects in the scene could be selected and manipulated using hand gestures.

Paper [11] gave an overview of the different basic gestures that users are most likely to use while manipulating 3D objects (As per user studies performed, it was observed that most users physically performed rudimentary object manipulation like zooming, scaling, stretching, and compressing the object which was supported effectively by this system developed). The Client-Server system setup was also demonstrated in [11]. The technicalities of depth and image recognition and tracking in 3D object manipulation and deploying AR applications on desktops by using webcams were observed. Additionally, the fact that gestures need to be as intuitive as possible to reduce mental strain for the end-users was also highlighted in [11].

Reference [12] explains some gesture recognition techniques from the perspective of developing an Augmented Reality media player application. It primarily focuses on an approach that is not computationally intensive, which is more suitable for an AR application. In the final approach, proposed by the authors, the focus is on recognizing the dynamic gestures of the user via a Webcam which could be a built-in or externally attached Camera. Various image processing algorithms like RGB to HSV conversion, blurring, thresholding, blob detection, etc have been integrated to analyse the gestures.

The approach used in [12] had no restrictions on the background of the image. It could handle dynamic gesture recognition via a live video feed, as the runtime processing time is negligible. Also, the colour model that is used is very strong. However, the Augmented Reality part was not dealt with in the paper and the complete integration was not defined. Additionally, [12] provided an overview of a gesture recognition system that is not computationally intensive and an AR Application-oriented approach to gesture recognition.

3. PROPOSED METHODOLOGY

To achieve the target of a generic and accessible AR application, the paper defines the scope of a demonstrable component as an android application that can reflect the activities running on a remote PC host. The android application, which can run on any simple smartphone (without the need for additional depth-sensing cameras) provides an Augmented Reality interface of the PC screen. The user can manipulate the AR interface (which helps avoid physical contact with any device) to operate on the PC remotely. The genericness is reflected in the fact that the applications running on the PC need not be AR compatible. Hence, absolutely any application that can run on the PC, would have an AR interface.

As seen in Figure 1, the approach taken to resolve this issue has two fundamental components. They are the interaction devices and the modes of interaction. There are three interaction components: the AR application, the remote host PC, and the gesture recognition server. The mode of interaction adopted in the chosen approach is a cloud-based interaction system.

3.1. Augmented Reality Application

The Augmented Reality (AR) Application acts as the controller of the entire system. In this part of the project, improvement in the accessibility of AR applications and their use is targeted. The application is deployable on a smartphone rather than a head-mounted device, due to its larger availability across the world. Also, it would only require a regular RGB camera that is built into a standard smartphone, without the necessity of a camera powered with pre-existing depth sensing technology. Since most AR applications use ARCore and ARKit which require smartphones with a depth-sensing camera, the project followed by this paper challenges this popular standard.

The design choice of choosing Unity [3] as the development engine over the Unreal Engine, was a conscious move, to make the AR application deployable to a broader spectrum of devices. However, this choice had to make a compromise on the larger feature set and the stronger and more robust engine that Unreal provides.

3.2. Remote Host

The remote host is the backend PC that is being manipulated by the AR application. The genericness of the system would be demonstrated in this part of the project, as any application visible on a PC screen could be replicated as a holographic AR Screen. The PC can hence be controlled via gestures (touchscreen gestures) in an AR application at any remote destination with a stable internet connection. For the demonstration, a browser application running on the PC is manipulated by the AR application.

3.3. Gesture Recognition Backend

The backend server runs the gesture recognition system. A design choice of using static gesture mapping is made so that the recognition can be quick. However, the challenge that this poses is the fact that not many gestures can be supported, however, the speed benefit outweighs the reduced number of gestures that are supported. Alternatively, an option writing the gesture recognizer in the AR application itself, limited the scope of gesture recognition, as it would have to run in the smartphone, which would be able to provide lower processing power [13]. Hence, an external server (which could also be the host PC itself), runs the gesture recognition system.

For the proof of concept of the methodology, the gesture recognition part involves 2 major parts - Hand segmentation and recognising the gesture in itself [14, 15]. For segmenting the hand from the surroundings, the major steps were background subtraction, thresholding and contour extraction. For recognizing the gestures, the steps involved were: Finding the convex hull and computing the extreme points, finding the centre of the palm and constructing a circle around the palm and fingers, performing bitwise AND between the hand region and the circle and determining the count or number of fingers. This was one of the various alternatives in which static gesture recognition could be implemented and can be easily modified according to the developer's requirements. Lots of research has gone into the field of gesture recognition, and some that are relevant in the current scenario include [16, 17, 18, 19].

3.4. Storage and Communication

The storage is handled using cloud storage systems. Other storage and communication alternatives included the use of a client-server system. But the client-server architecture restricted the AR application to run on the same network as the host PC. This was a clear disadvantage. Hence the choice of using a cloud storage system was made clear. However, this choice had the drawback of being a little slower than its counterpart.

The implementation used Google's Firebase Cloud System [20] for all the storage requirements. The Firebase Real-Time Database (RTDB) was used to store the recognized gesture [21]. The Firebase RTDB is a cloud-based NoSQL database that syncs data in real-time [22, 23], hence it was apt in storing the gesture identified by the gesture recognizer backend, to be used immediately by the remote host to update its state. The Firebase Cloud Storage is used to store the host PC's current state (for updating the AR environment) and the AR environment (for gesture recognition). This Cloud Storage unit of Firebase is a scalable cloud infrastructure [24] capable of storing large binary objects as blobs. The states that are saved in the form of screenshot images are hence stored and retrieved quickly, efficiently, and securely, in the Firebase Cloud Storage.

3.5. Overall System Design

The overall flow of the proposed methodology proceeds as illustrated by the sequence diagram in Figure 1. The process begins with the user enabling the host PC to be accessed by the Augmented Reality application. This is automatically followed by the storage of the initial state (screenshot of the entire PC screen) in the Firebase Cloud Storage. Once the user opens the AR application via the user's smartphone remotely, the state of the host PC is fetched from the Firebase Cloud Storage and rendered as an AR interface in the mobile application. As the user manoeuvres through the AR interface, the state (screenshot) of the AR environment is also stored in the Firebase Cloud Storage. This picture/state of the AR environment is constantly monitored by the gesture recognizer to identify the user's gestures. Once the gesture is identified, it is updated in the Firebase Real-Time database. The host PC reads the gesture from the Firebase Real-Time Database and updates the state of the remote PC. This new state is updated in the Firebase Cloud Storage, which is reflected in the AR application. And the process keeps repeating.

4. RESULTS AND CONCLUSION

The proposed methodology was applied on a browser (Google Chrome, Microsoft Edge as seen in Figure 2) that runs on a laptop with an internet connection. The browser could be accessed remotely via the Augmented Reality interface in an Android application that was connected to the internet, on a separate network. The syncing had minor lags depending on the speed of the internet connection on both the devices (smartphone and remote host PC). However, this

proposed methodology gives a simple, generic, and highly accessible solution for Augmented Reality interfacing for any application.

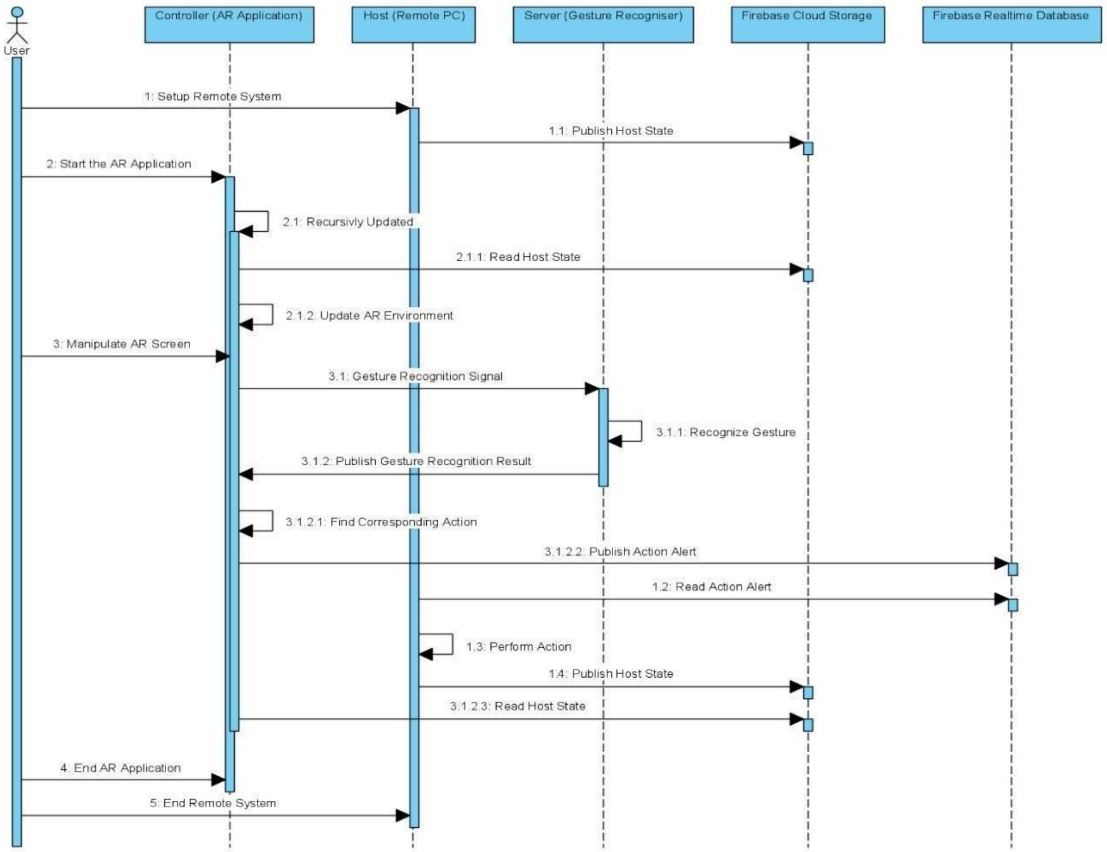

Figure 1. Sequence diagram of the proposed approach

5. FUTURE SCOPE

The methodology proposed in this paper can be further extended to thorough implementation. If funds permit, the Augmented Reality application can be extended to greater graphical quality (with holographic effects, and 3D rendering). Also, a wider range of gestures can be supported by a developer who wishes to extend its functionality. Moreover, there is new scope for innovation in the methodology itself, by taking the platform to an offline access system. This would not only improve the accessibility of the system but would also reduce the lags caused due to internet connectivity issues.

Figure. 2. A view of the AR Application (controlling Microsoft Edge remotely) on a smartphone in a basic demonstration. Only the quadrilateral in the middle is the AR rendering. The remaining is the user's hand and the real environment/surroundings behind.

REFERENCES

[1] Julie Carmigniani et al. "Augmented reality technologies, systems and applications". In: *Multimedia tools and applications* 51.1 (2011), pp. 341–377.

[2] Kevin Bonsor amp; Nathan Chandler. *How Augmented Reality Works*. Feb. 2001. URL: https : / / computer . howstuffworks.com/augmented-reality.html.

[3] Vinh T Nguyen and Tommy Dang. "Setting up virtual reality and augmented reality learning environment in unity". In: *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. IEEE. 2017, pp. 315–320.

[4] Daria Dubrova. Sept. 2018. URL: https : / / theappsolutions . com / blog / development / augmented reality-challenges/.

[5] S Siji Rani, KJ Dhrisya, and M Ahalyadas. "Hand gesture control of virtual object in augmented reality". In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2017, pp. 1500–1505.

[6] *Gesture Interaction for Consumer Devices*. URL: https: //www.crunchfish.com/how- does- the-world- look through-smart-glasses/augmented-reality/.

[7] Rafiqul Zaman Khan and Noor Adnan Ibraheem. “Hand gesture recognition: a literature review”. In: *International journal of artificial Intelligence & Applications* 3.4 (2012), p. 161.
[8] Rodrigo Silva, Jauvane C Oliveira, and Gilson A Giraldi. “Introduction to augmented reality”. In: *National laboratory for scientific computation* 11 (2003).
[9] John Aliprantis et al. “Natural Interaction in Augmented Reality Context.” In: *VIPERC@ IRCDL*. 2019, pp. 50–61.
[10] Huidong Bai et al. “Markerless 3D gesture-based interaction for handheld augmented reality interfaces”. In: *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2013, pp. 1–6.
[11] Mark Billinghurst, Tham Piumsomboon, and Huidong Bai. “Hands in space: Gesture interaction with augmented-reality interfaces”. In: *IEEE computer graphics and applications* 34.1 (2014), pp. 77–80.
[12] Sandeep Vasave and Amol Plave. “Study of Gesture Recognition methods and augmented reality”. In: *arXiv preprint arXiv:1411.5137* (2014).
[13] Arnaud Lemoine et al. “Hand gesture recognition system and method”. In: *Patent US* 6128003 (2000).
[14] Gogul Ilango. *Hand Gesture Recognition using Python and OpenCV - Part 1*. Apr. 2017. URL: https://gogul. dev/software/hand-gesture-recognition-p1.
[15] Gogul Ilango. *Hand Gesture Recognition using Python and OpenCV - Part 2*. Apr. 2017. URL: https://gogul. dev/software/hand-gesture-recognition-p2.
[16] Gur Raunaq Singh. *Introduction to Using OpenCV With Unity*. URL: https://www.raywenderlich.com/5475introduction-to-using-opencv-with-unity.
[17] M Naveenkumar and A Vadivel. “OpenCV for Computer Vision Applications”. In: *Proceedings of National Conference on Big Data and Cloud Computing (NCBDC’15)*. 2015.
[18] Prasham Parikh. *Google Open Sources Real-Time Hand Gesture Recognition Algorithm For Developers*. Aug. 2019. URL: https://in.mashable.com/tech/6130/googleopen- sources- real- time- hand- gesture- recognitionalgorithm-for-developers.
[19] Siddharth S Rautaray. “Real time hand gesture recognition system for dynamic applications”. In: *International Journal of UbiComp (IJU)* 3.1 (2012).
[20] Laurence Moroney, Moroney, and Anglin. *Definitive Guide to Firebase*. Springer, 2017.
[21] Wu-Jeng Li et al. “JustIoT Internet of Things based on the Firebase real-time database”. In: *2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)*. IEEE. 2018, pp. 43–47.
[22] Laurence Moroney. “The firebase realtime database”. In: *The Definitive Guide to Firebase*. Springer, 2017, pp. 51–71.
[23] Baudoux Nicolas and Bauwin Lucie. “Real-Time database: Firebase INFO-H-415: Advanced database”. In: *Universite Libre de Bruxelles. Academic year´* 2018 (2017).
[24] Laurence Moroney. “Cloud storage for firebase”. In: *The Definitive Guide to Firebase*. Springer, 2017, pp. 73–92.

AUTHORS

Arya Rajiv Chaloli:
Arya completed her BTech in Computer Science and Engineering (CSE) at PES University, Bangalore. Currently works at Microsoft India, Hyderabad.

K Anjali Kamath:
Anjali completed her Bachelor of Technology in Computer Science and Engineering (CSE) from PES University, Bangalore. Working as a Software Engineer in Citrix Systems, Bangalore.

Divya T Puranam:
Divya completed her BTech in Computer Science and Engineering from PES University, Bangalore. Works as a software engineer at Cisco Systems, Bangalore.