# AR / VR Keyboard Interface

GUIDED BY:

Dr. Thanasekhar B,
Associate Professor,
MIT Campus,
Anna University.

TEAM MEMBERS:

1. Hariprasanth M S (2018503030)
2. Chandhru R (2018503512)
3. Venuprasath P (2018503070)

## INTRODUCTION

- Augmented reality (AR) delivers immersive and engaging experiences that seamlessly blend virtual objects with the real world.
- Augmented reality is used to enhance natural environments or situations and offer perceptually enriched experiences.
- The goal of AR is to enhance a user’s perception of the real world. In a touchless system, without any sort of physical interface the user will rely almost entirely on visual cues when interacting with the system.
- Augmentation techniques are typically performed in real time and in semantic contexts with environmental elements.
- Immersive perceptual information is sometimes combined with supplemental information like scores over a live video feed.
- The keyboard object acts sole component of the user interface. The development of an immersive, intuitive virtual keyboard interface is paramount to the success and usability of this system.

## PROBLEM DEFINITION

- When using Augmented Reality apps, we need an interface for inputting.
- Traditionally keyboard and mouse are used.
- But in AR environments, the use of dedicated keyboard and mouse may need additional space which limits AR experience.
- Instead of using a physical keyboard, we propose a vision based gesture-controlled Augmented Reality keyboard using mobile device.
- Our proposed keyboard has poke-typing feature.
- We propose the keyboard for both Augmented and Virtual environments.
- This provides additional security to the user as the keyboard will only be visible to the user through the head mounted device (HMD).

## Literature Survey

- Lee et al in 2003 [1], proposed an Augmented Virtual Reality Keyboard using fiducial markers and colour markers detect and track keyboard and fingertips.
- Zhang, Yinda et al in 2020 [2], used Dual-camera and Dual-pixel sensors to estimate the depth of the object in the image and this hardware is available in most modern smartphones.
- Jung et al in 2014 [6], proposed a novel virtual keyboard interface 'Augmented Keyboard' for a smart glasses.
- An intuitive and easy-to-use interaction design is proposed in order to inherit the uses in QWERTY keyboard by employing the two features of hand: hand orientation and hand pose.
- Manomotion SDK [10] provides hand tracking features on Android and iOS devices and it enables real-time 2D/3D hand tracking and gesture control.

## Literature Survey (Contd.)

- Sam Green in 2016 [3], implements a system for hands free typing using swipe gestures to recognise words.
- Background noises are eliminated with the help of initially captured frames and using openCV library they detect fingers and assign suitable gestures for specific actions
- From the obtained set of letters, they run an algorithm (Levenshtein distance) to predict a set of words from a dictionary which we can make use of to predict words with good accuracy for the swype feature.
- Chen et al in 2019 [9], proposed word-gesture typing using 6 Degree of Freedom VR controllers.
- The gestures are captured and an algorithm is used by them [9] to predict the word.
- This [9] is implemented in Unity3D. Which is similar to swype keyboard.

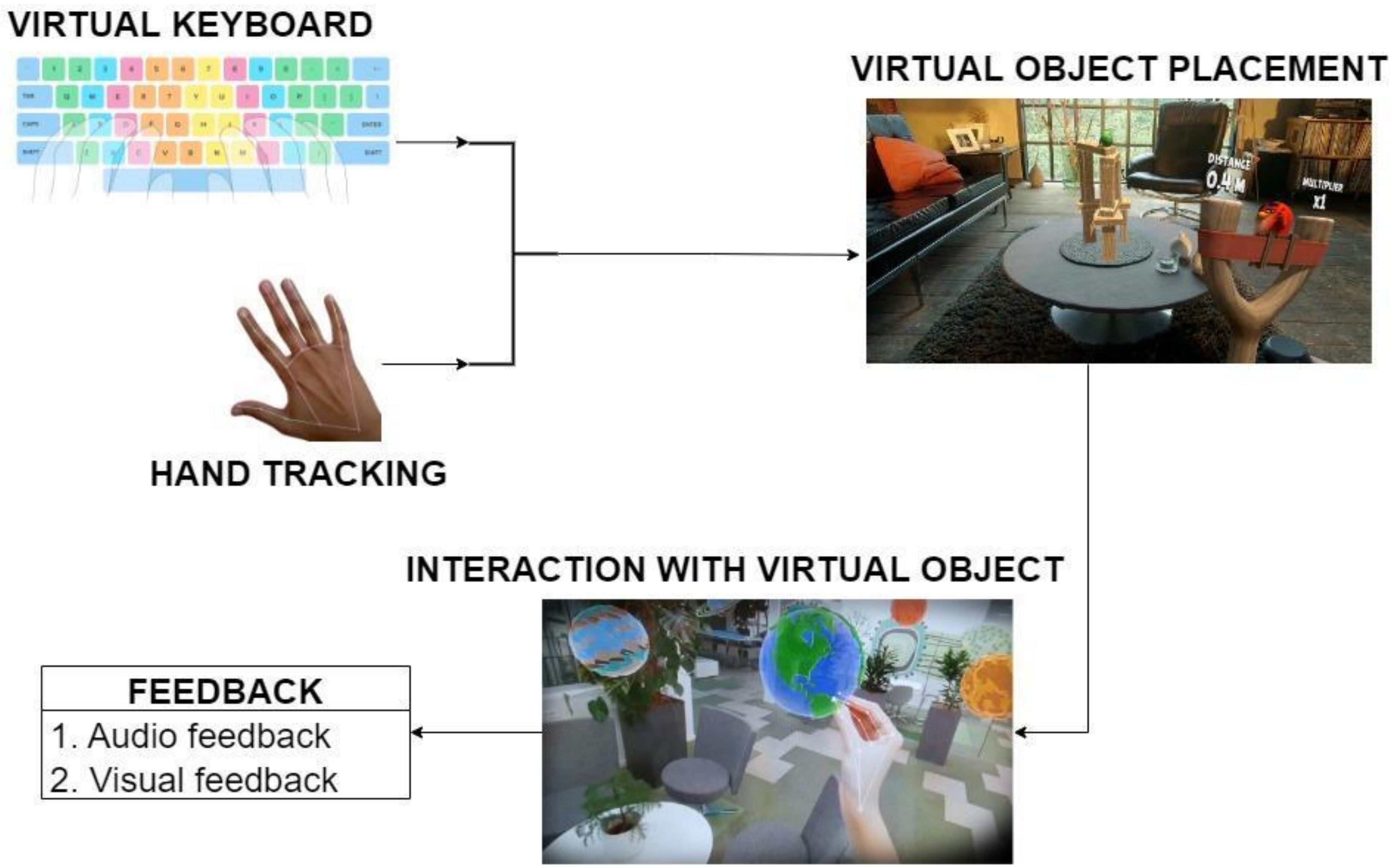
## Literature Survey (Contd.)

- A.Maiti et al in 2017 [4], used a external device (EPSON Moverio BT-200) which superimposes the keyboard over physical keyboard and establishes a temporary secure wireless link with the computer
- This secure link can be established using widely available wireless technologies, such as Bluetooth, and made secure using symmetric encryption schemes such as AES.
- L. H. Lee et al in 2019 [5] used Microsoft's hololens and designed HIBEY. The main advantage of this methodology is the minimal usage of available space for projecting the augmented keyboard.
- (1.)Fast-forward zone for Selecting key, (2.)preparation zone for confirming key (3.) recall zone for Backspace to withdraw the selected character.
- In Addition to that word prediction is also done to precomplete the required word. All the selection processes are done with the help of specific hand gestures.
- This methodology can be used when accurate key interactions don't work out well.

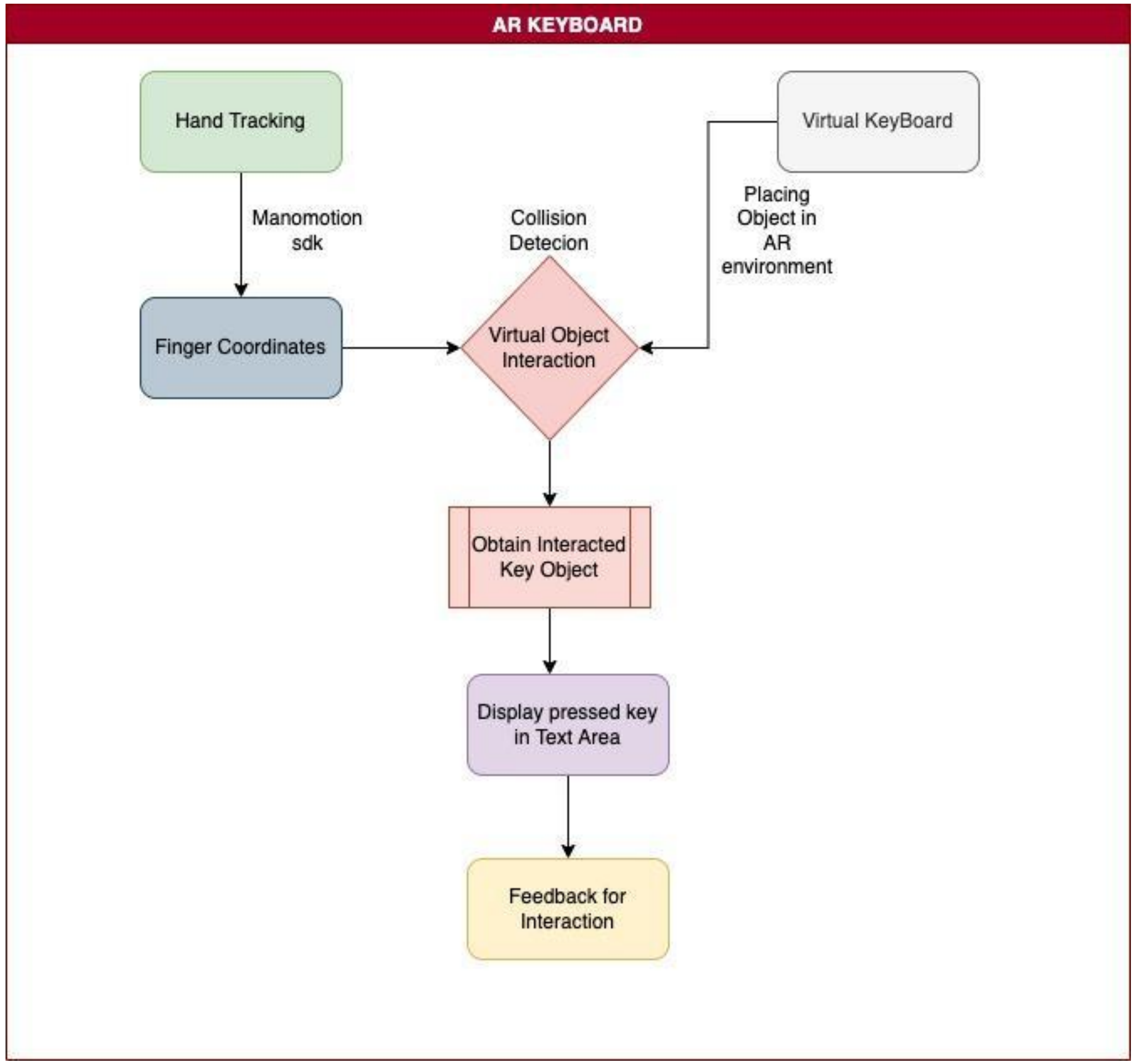
## Literature Survey (Contd.)

- MusiKeys, 2021 [7], focuses mainly on auditory feedback to mimic physical keyboard keypress.
- In Oculus Quest 2, tracking is very precise and consistent for the thumbs, index, and middle fingers, but it cannot reliably track the ring finger.
- The system keyboard on Microsoft HoloLens 2, provides auditory feedback on keypress and release, however, only allows the index finger to type.
- Habibi in 2021 [8], provided a background study of AR text entry.
- He emphasised on the importance of sound to detect a click or swipe on a surface.
- This paper [8] used a mic attached to the surface of interaction to detect whether a click operation was performed by using the tap sound.
- They also studied swipe typing and provided results on the usability and performance of those methods.

# MIND MAP


VIRTUAL KEYBOARD
HAND TRACKING
VIRTUAL OBJECT PLACEMENT
DISTANCE 0.4 M
MULTIPLIER x1
INTERACTION WITH VIRTUAL OBJECT
FEEDBACK
1. Audio feedback
2. Visual feedback

# SYSTEM ARCHITECTURE


AR KEYBOARD
Hand Tracking
Manomotion sdk
Finger Coordinates
Collision Detecion
Virtual Object Interaction
Virtual KeyBoard
Placing Object in AR environment
Obtain Interacted Key Object
Display pressed key in Text Area
Feedback for Interaction

# MODULE 1 - Hand Tracking

**Hand Tracking**

1. Import Manomotion SDK pro into Unity project
2. Include the prefabs ManomotionManager, ManomotionCanvas, ManoVisualization and SkeletonManager into the Scene
3. Set the liscence key in ManomotionManager
4. Set the camera as the AR camera in ManomotionManager and ManoVisualization
5. Use the 3D Joints Collider as the Joint Prefab

**Input:**

Live video with full hand (upto ankle ) visible in video.

**Outcome:**

The joints (x, y ,z) coordinates are provided with individual positions for each of the 21 joints of the hand skeleton.

## MODULE 1 - Hand Tracking

**Script :**

```
private void SkeletonModel(int modelToLoad, int previousModel)
{
    if (jointPrefab[modelToLoad].transform.childCount == jointsLength)
    {
        _listOfJoints.Clear();
        jointPrefab[previousModel].SetActive(false);
        jointPrefab[modelToLoad].SetActive(true);
        for (int i = 0; i < jointPrefab[modelToLoad].transform.childCount; i++)
        {
            _listOfJoints.Add(jointPrefab[modelToLoad].transform.GetChild(i).gameObject);
        }
        lineRenderers = new LineRenderer[6];
        lineRenderers = (jointPrefab[modelToLoad].GetComponentsInChildren<LineRenderer>());
        ResetLineRenderers();
    }
    else
    {
                Debug.LogFormat("Current  model  have  {0}  joints,  need  to  have  21  joints",
jointPrefab[modelToLoad].transform.childCount);
    }
}
```

# MODULE 2 - Keyboard Layout

**Keyboard design**

1. Create an empty gameobject (keyboard).
2. Make a plane object as a child of it which is used as a housing for the keyboard.
3. Add the required keys (cube objects) as the child of keyboard object.
4. Add a canvas to display the entered text.
5. Add the necessary Script in Keyboard and Interactable button so that when a key is pressed, the canvas displays the entered text.

**Outcome:**

A virtual keyboard prefab which can be used as an reusable object

# MODULE 2 - Keyboard Layout

**Scripts :**

```
public class KeyBoard : MonoBehaviour
{
    const string BACKSPACE = "backspace";
    private string userInput = "";
    private TMP_Text inputBuffer;

    private void Start() {
        userInput = "";
        inputBuffer = (TMP_Text)GameObject.Find("InputBuffer").GetComponent<TMP_Text>();
        inputBuffer.text = "";
    }

    public void ButtonClicked(string input)
    {
        if(string.Compare(BACKSPACE, input)==0)
        {
            userInput = userInput.Remove(userInput.Length - 1);
        }
        else
            userInput += input;
        inputBuffer.text = userInput;
        Debug.Log("Keypad input: "+ inputBuffer.text);
    }
}
```

## MODULE 3 - Keyboard Interaction

**Interaction tracking**

1. Add a collider object to the target positions in the hand skeleton.
2. Add rigid body and box collider object to the interactable button.
3. Set the isTrigger field in Box collider component as true in Interactable button.
4. Set the IsKinematic field as true on Rigid body component in the button.
5. Add the AR Keyboard Button script to the button object.
6. Drag and drop the materials for the AR cube in the inspector section.
7. Set the parent object field of Interactable button as the keyboard object.

**Input:**

Tracked hand finger tip coordinates , Augmented object coordinates

**Outcome:**

Key Press Detection

## MODULE 3 - Keyboard Interaction

**Scripts :**

```
private void OnTriggerEnter(Collider other)
{
    Debug.Log(other.gameObject.name);
    if(other.gameObject.name == indexFingerTip)
    {
        triggerTime = Time.time;
        cubeRenderer.sharedMaterial = arCubeMaterial[1];
        Debug.Log("Outer Button entered");
    }
}

private void OnTriggerExit(Collider other)
{
    Debug.Log("Button released");
    triggerTime = Time.time - triggerTime;
    if(triggerTime > KEYPRESS_DELAY && other.gameObject.name == indexFingerTip)
    {
        keyboardClicked.Invoke();
        Handheld.Vibrate();
    }
    Debug.Log("Trigger time: " + triggerTime);
    cubeRenderer.sharedMaterial = arCubeMaterial[0];
}
```

## MODULE 4 - VR Space

**Designing VR environment:**

1. Add a cube object and set the textures and materials for the surface.
2. Include some objects in the room like cubes, spheres and other objects.
3. Add the CardboardStartup script to the cube room gameobject.
4. Place the object in the scene.

**Prerequisite:**

- Tracked hand finger tip coordinates
- Augmented object coordinates

**Outcome:**

- A Virtual Interactable keyboard in VR space

# MODULE 4 - VR Space

**Scripts :**

```
public class CardboardStartup : MonoBehaviour
{
    public void Start()
    {
        Screen.sleepTimeout = 
SleepTimeout.NeverSleep;
        Screen.brightness = 1.0f;
        if (!Api.HasDeviceParams())
        {
            Api.ScanDeviceParams();
        }
    }
```

```
    public void Update()
    {
        if (Api.IsGearButtonPressed)
        {
            Api.ScanDeviceParams();
        }

        if (Api.IsCloseButtonPressed)
        {
            Application.Quit();
        }

        if (Api.IsTriggerHeldPressed)
        {
            Api.Recenter();
        }

        if (Api.HasNewDeviceParams())
        {
            Api.ReloadDeviceParams();
        }

        Api.UpdateScreenParams();
    }
}
```

## MODULE 5 - Feedback & Testing

**Input:** Keyboard model

**Outcome:** Improved user experience with audio and visual feedback. Performance report.

**Scripts :**

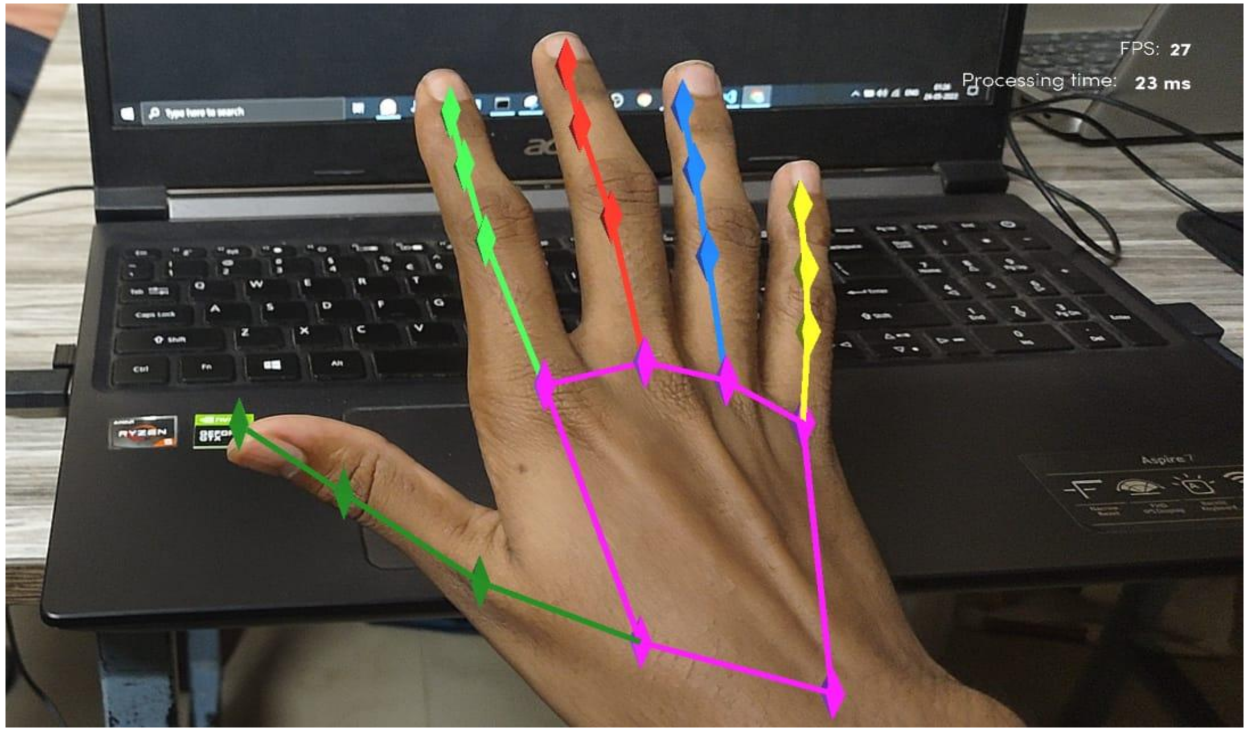
```
private void Update()
{
       if  (ManomotionManager.Instance.Hand_infos[0].hand_info.tracking_info.skeleton.confidence  <
skeletonConfidenceThreshold)
    {
        cubeRenderer.sharedMaterial = arCubeMaterial[0];
    }
}

private void OnTriggerStay(Collider other)
{
    if (other.gameObject.name == indexFingerTip)
    {
        cubeRenderer.sharedMaterial = arCubeMaterial[1];
    }
}
```

# IMPLEMENTATION DETAILS

- Unity 2020.3.32f is used for the application development.
- Hand is tracked using Manomotion SDK pro, which internally uses machine learning models as well as the ARCore library available in modern smartphones.
- The skeleton manager provided by Manomotion SDK is modified for our application to provide interaction feature.
- The confidence level for the skeleton is used as 0.01f.
- A QWERTY keyboard is designed in Unity.
- Interaction feature is provided to the buttons in keyboard.
- We have used both performance of device and user as our evaluation metric.
- We developed 2 applications which include AR and VR.

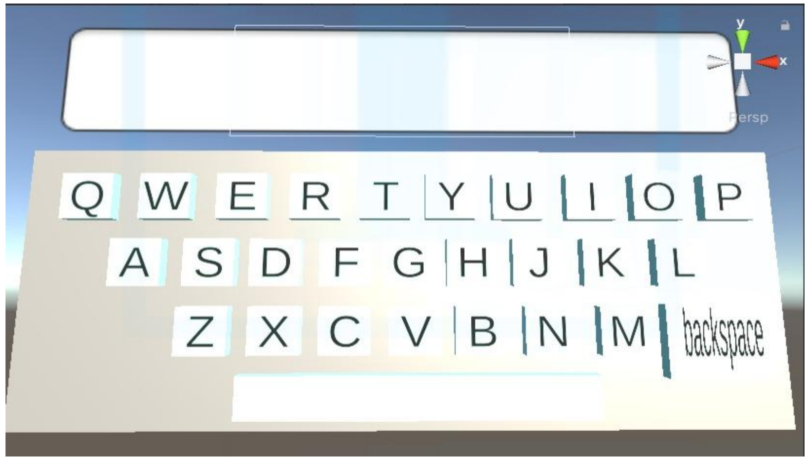
# EXPERIMENTAL RESULTS

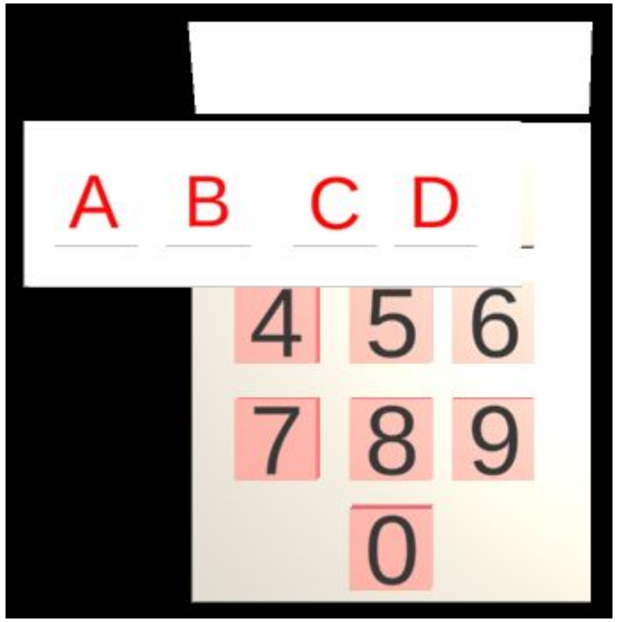
**Hand Tracking**



# EXPERIMENTAL RESULTS

**Keyboard Layouts**

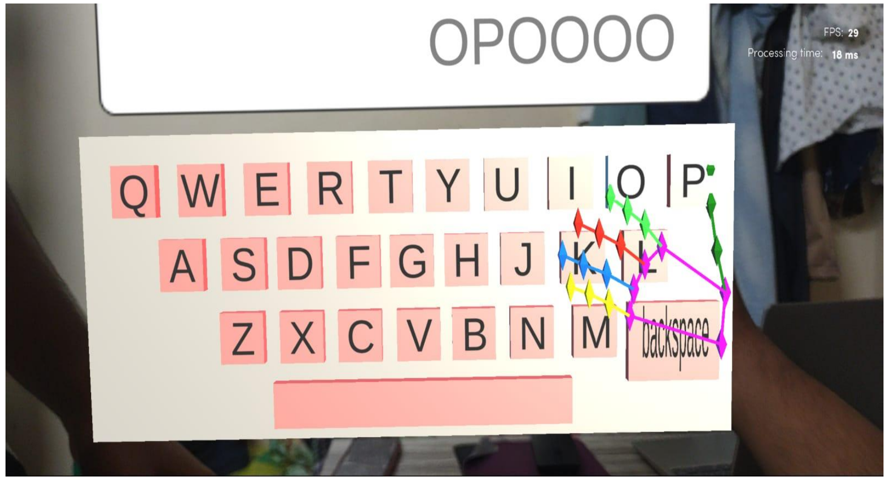




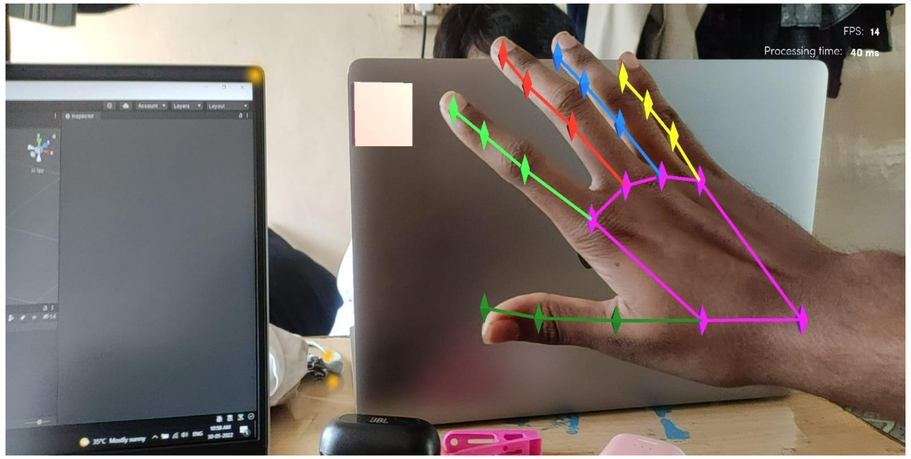
# EXPERIMENTAL RESULTS

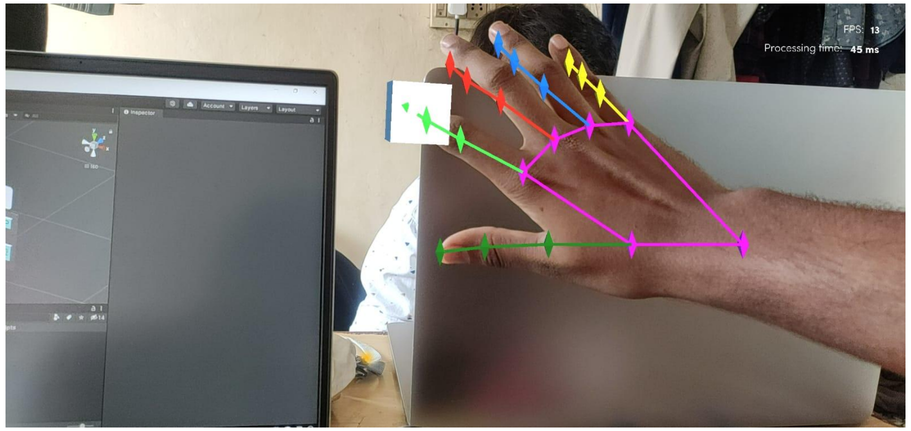
**KeyBoard Interaction**

Typed word



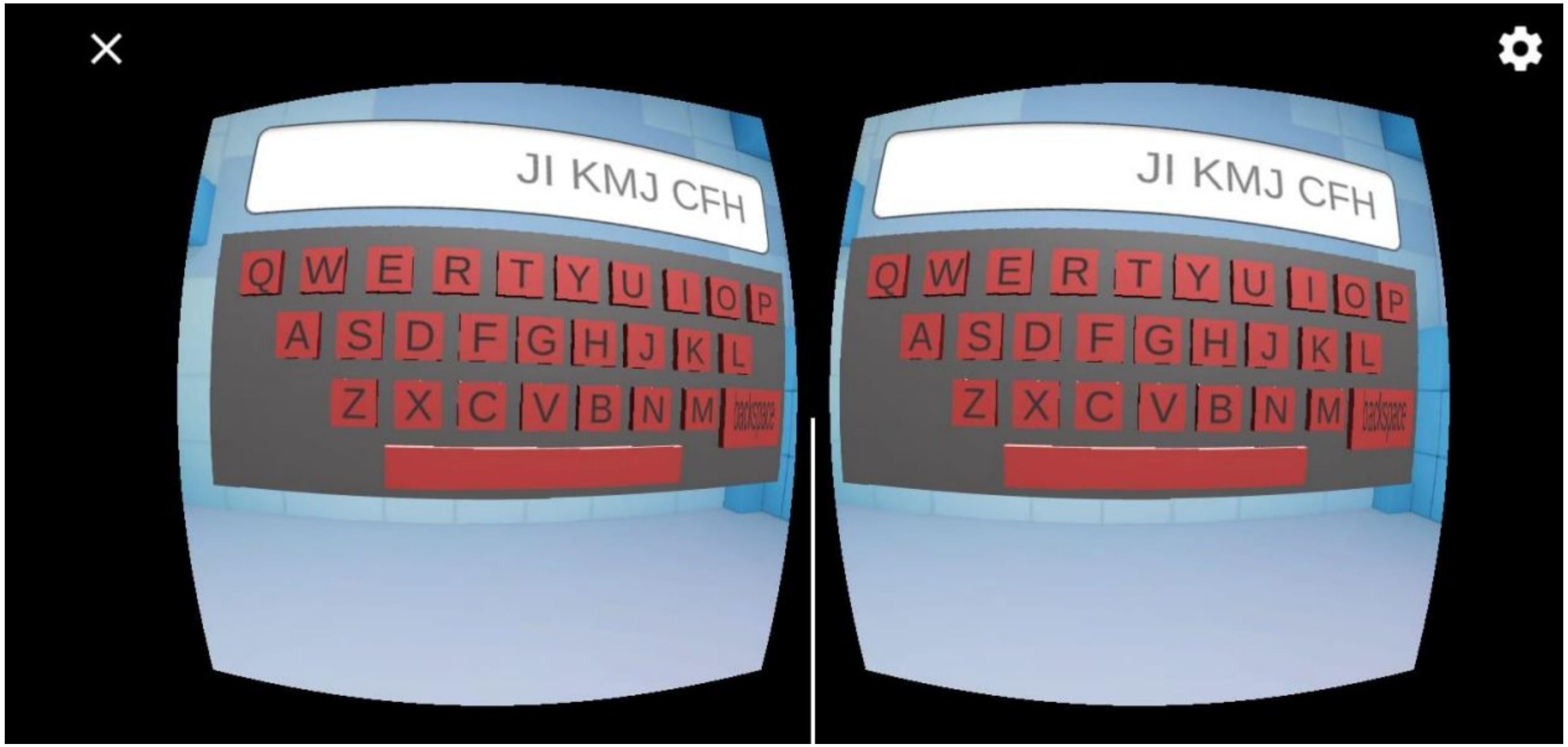
Before and after interaction





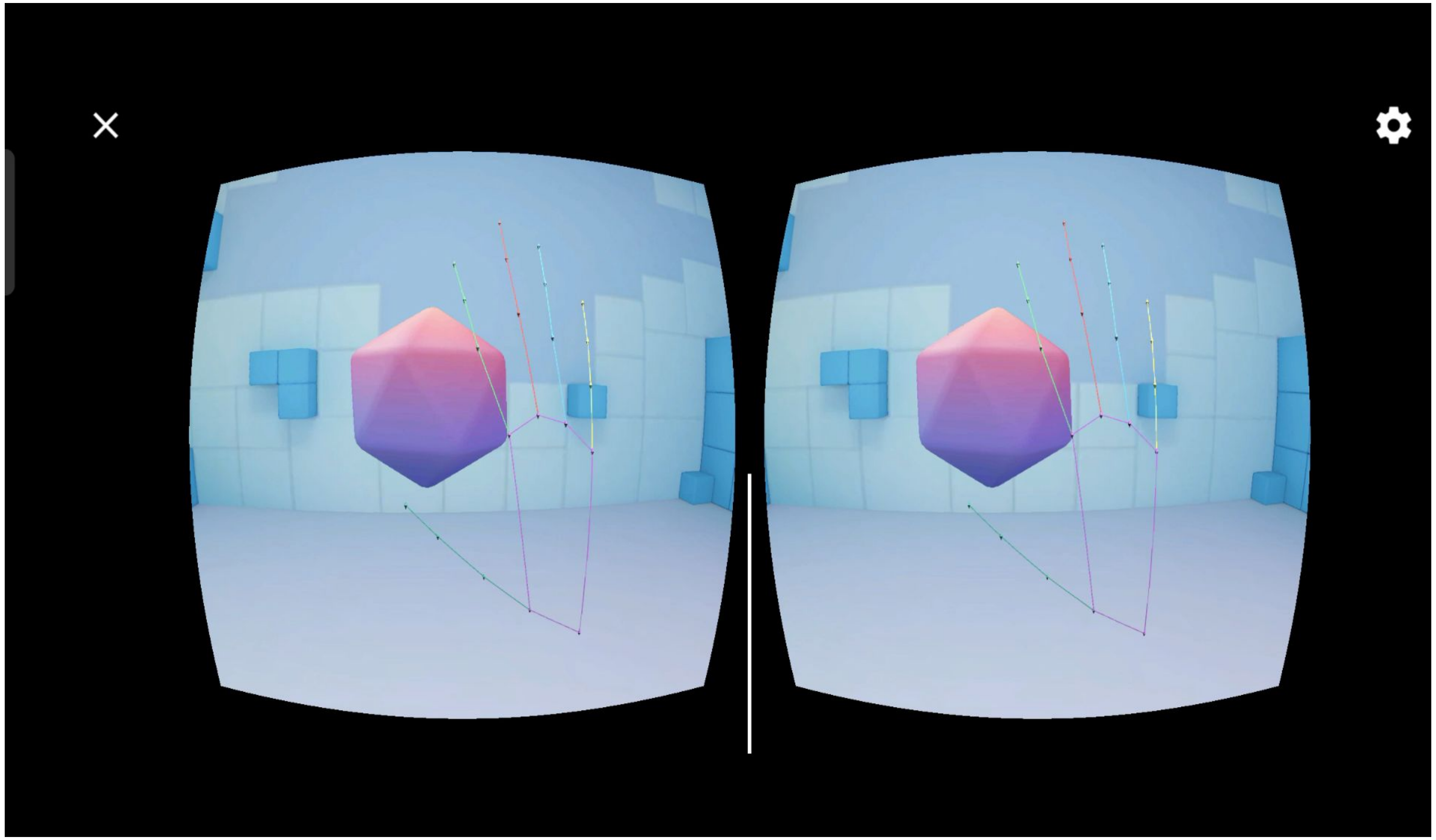
# EXPERIMENTAL RESULTS

**Keyboard in VR Space**



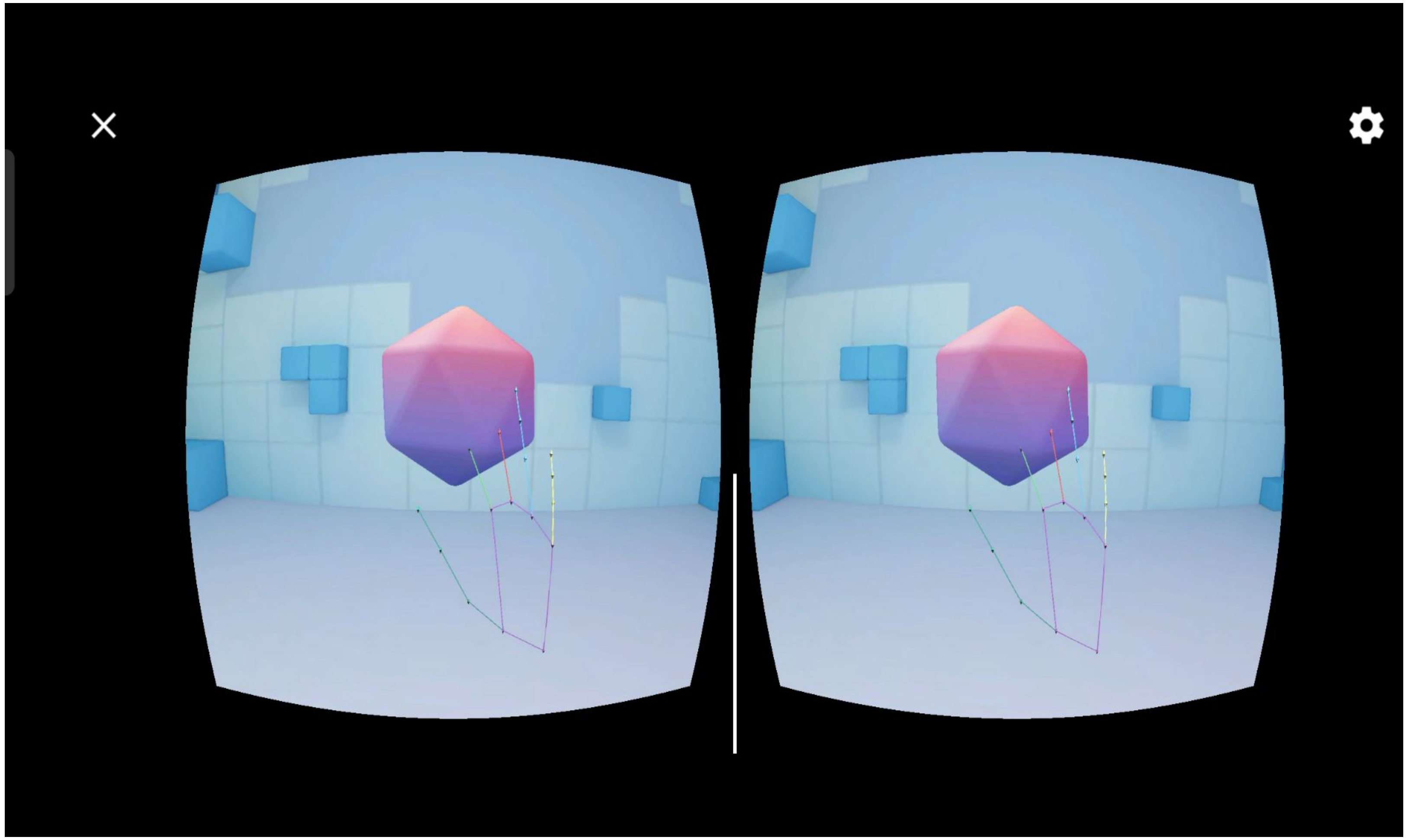
# EXPERIMENTAL RESULTS

**VR object before interaction**



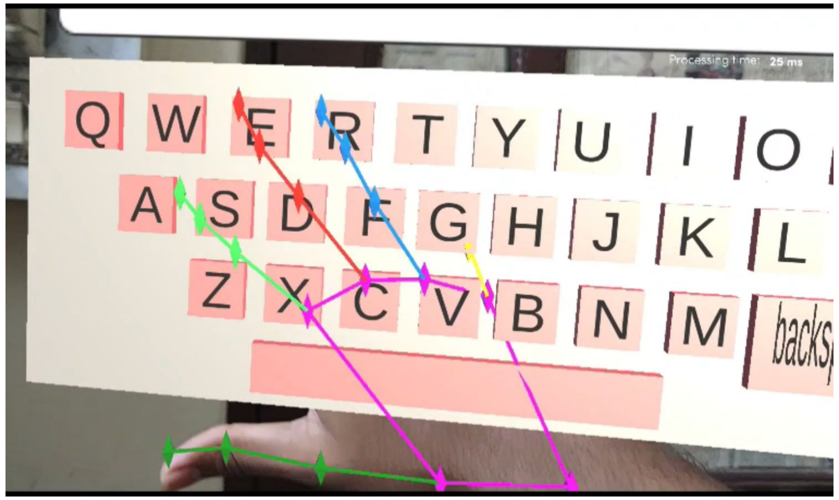
# EXPERIMENTAL RESULTS

**VR object after interaction**

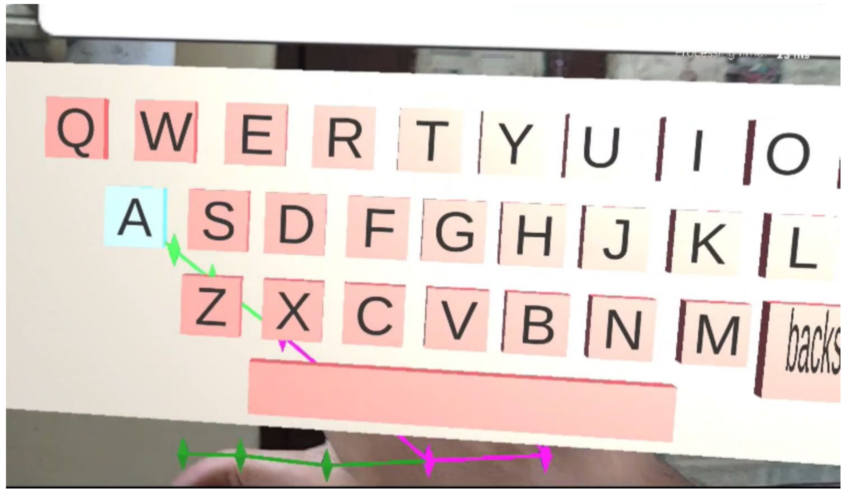


# EXPERIMENTAL RESULTS

## Keyboard in AR Space



Keyboard without visual feedback



Keyboard with visual feedback

# RESULT ANALYSIS

**METRICS USED TO EVALUATE THE SYSTEM:**

1) FPS in mobile device
   a) Frames the application renders per second

2) Characters typed per minute
   a) The average number of characters typed in a minute by a user

3) Error rate
   a) (Incorrectly typed characters / Total number of typed characters) * 100

## RESULT ANALYSIS

**Experimental Setup 1:**

We had a total of three participants who were able to use the AR keyboard for up to 10 minutes for practice before we observed them typing 43 predetermined letters.

**Feedbacks from Result 1:**

1. When a key is pressed, then the nearby character is pressed without user intention.

2. The hand tracked by the Manomotion SDK stops when some portion of the hand gets out of the field of view of the camera.

3. The vertical keyboard didn't feel like a natural scenario.

4. When using the application with different backgrounds, the SLAM module loses the keyboard position and it moves.

## RESULT ANALYSIS

**Result 1 (Tested on OnePlus Nord 2)**

Input: The quick brown fox jumps over the lazy dog

| **Participant** | **Error Rate** | **Speed (CPM)** |
|---|---|---|
| **User 1** | 0.25 | 10 |
| **User 2** | 0.27 | 14 |
| **User 3** | 0.18 | 11 |

# RESULT ANALYSIS

**Experimental Setup 2:**

- Next, we ran a second user study, this time focusing on error rate and typing speed, which are standard metrics for evaluating VR keyboards.
- We used the same participants and the same set of inputs for them.
- Here we increased the time delay between the keypress, so that nearby keys wont get pressed as the user takes their hand from the pressed key.
- Also the size of the keyboard is reduced so that the user can see a considerable portion of the scene and keep their hands within the field of view of the camera.
- And the rotation of the keyboard is changed to 15 degrees to provide a realistic feel.

## RESULT ANALYSIS

**Result 2 (Tested on OnePlus Nord 2)**

Input: The quick brown fox jumps over the lazy dog

| **Participant** | **Error Rate** | **Speed (CPM)** |
|---|---|---|
| **User 1** | 0.20 | 17 |
| **User 2** | 0.23 | 22 |
| **User 3** | 0.17 | 20 |

## CONCLUSIONS

- AR keyboard is a touchless virtual keyboard in an augmented reality environment.
- This system enables the user to type mid air by poke typing, enabling a natural way to interface with computers using our hands rather than a physical keyboard component.
- The user study conducted has shown that, the system cannot yet compete with the performance of physical keyboards in terms of speed and accuracy.
- But it provides a viable framework for mid-air text entry in which users do not need to touch any physical surfaces and also demonstrates the suitability of augmented reality as an interface paradigm for deviceless interface systems.
- This keyboard can be used for short text entry tasks.

## FUTURE WORK

- Currently the accuracy of the characters typed is low (75%).
- This can be improved by adding a word prediction system which can significantly improve the words per minute metric.
- Comparison should be made by setting different values for cooldown time between each keypress, keypress time, smoothness value for hand skeleton and the best parameter should be tested with a number of users.
- Integrating the Augmented reality keyboard in Google cardboard or similar setup would help users to easily work with the system.

# References

1. Lee, M. and Woo, W., 2003, December. ARKB: 3D vision-based Augmented Reality Keyboard. In ICAT.
2. Zhang, Y., Wadhwa, N., Orts-Escolano, S., Häne, C., Fanello, S. and Garg, R., 2020, August. Du 2 net: Learning depth estimation from dual-cameras and dual-pixels. In European Conference on Computer Vision (pp. 582-598). Springer, Cham.
3. GREEN, S., 2016. Augmented Reality Keyboard for an Interactive Gesture Recognition System (Doctoral dissertation, School Of Computer Science & Statistics Submitted to the University of Dublin, Trinity College).
4. Maiti, A., Jadliwala, M. and Weber, C., 2017, March. Preventing shoulder surfing using randomized augmented reality keyboards. In 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) (pp. 630-635). IEEE.
5. Lee, L.H., Lam, K.Y., Yau, Y.P., Braud, T. and Hui, P., 2019, March. Hibey: Hide the keyboard in augmented reality. In 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom (pp. 1-10). IEEE.
6. Jung, J., Jeon, J., Lee, H., Kwon, K., Zemerly, J. and Yang, H.S., 2014, November. Augmented keyboard: a virtual keyboard interface for smart glasses. In Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (pp. 159-164).
7. Krasner, A., Gabbard, J.L. and Burnett, G., 2021, October. MusiKeys: Investigating Auditory-Physical Feedback Replacement Technique for Mid-air Typing. In 2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct) (pp. 510-512). IEEE.
8. Habibi, R., 2021. Detecting surface interactions via a wearable microphone to improve augmented reality text entry (Doctoral dissertation, Michigan Technological University).
9. Chen, S., Wang, J., Guerra, S., Mittal, N. and Prakkamakul, S., 2019, May. Exploring word-gesture text entry techniques in virtual reality. In Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (pp. 1-6).
10. Manomotion SDK (https://www.manomotion.com/ Last accessed on 8th June 2022)