Multi-user Multi-platform xR collaboration: System and evaluation

Johannes Tümler[1][0000−0002−4788−2667], Alp Toprak[1], and Baixuan Yan[1]

Anhalt University of Applied Sciences, Köthen, Germany
johannes.tuemler@hs-anhalt.de, alptoprak94@gmail.com,
baixuanyan.sam@gmail.com

Abstract. Virtual technologies (AR/VR/MR, subsumed as xR) are used in many commercial applications, such as automotive development, medical training, architectural planning, teaching and many more. Usually, existing software products offer either VR, AR or a 2D monitor experience. This limitation can be a hindrance. Let's draw a simple application example: Users at a university shall join an xR teaching session in a mechanical engineering lecture. They bring their own xR device, join the session and experience the lecture with xR support. But users may ask themselves: Does the choice of my own xR device limit my learning success?
In order to investigate multi-platform xR experiences, a software framework was developed and is presented here. This allows one shared xR experience for users of AR smartphones, AR/MR glasses and VR-PCs. The aim is to use this framework to study differences between the platforms and to be able to research for better quality multi-user multi-platform xR experiences.
We present results of a first study that made use of our framework. We compared user experience, perceived usefulness and perceived ease of use between three different xR device types in a multi-user experience. Results are presented and discussed.

Keywords: Augmented Reality · Virtual Reality · Multi-User · Collaboration · Metaverse

1 Motivation and Goals

In the last 5-10 years, virtual technologies (AR/VR/MR, subsumed as xR) have matured from exotic expert tools to valuable industry standard. Sample commercial application fields are automotive development, medical training, architectural planning, teaching and many more [8,13,4,7].

The trend goes towards collaborative multi-user xR scenarios: Software allows synchronizing data between xR devices through local networks or cloud services[1].

[1] Examples: Microsoft Azure Spatial Anchors (https://azure.microsoft.com/en-us/services/spatial-anchors/), Vive Sync (https://sync.vive.com/), Mozilla Hubs (https://hubs.mozilla.com/)

The final authenticated version is available online at https://doi.org/10.1007/978-3-031-05939-1_6.

Social platforms and large companies propagate this as the "Metaverse": The next big step in xR. It shall be used to leverage social interaction and spatial cooperation in all kinds of xR use cases. Therefore, let us assume the following two sample use cases:

#1 University students physically join a lecture for mechanical engineering. The professor activates the xR session. Each student can join that session with their own xR devices (from Google Cardboard VR via handheld smartphone AR to see-through AR glasses). The teacher controls what the students can see, where virtual objects are located in the scene and which virtual objects can be used for interaction. If the students can't join the lecture hall ("co-location"), they can also join remotely using VR.

#2 The second application is a field engineer that needs to service a special purpose machine. He could use a mobile AR device (handheld, head-worn) to get servicing instructions. Meanwhile a backoffice engineer joins that servicing session using a VR device looking at the same machine (full virtual data). Both collaborate on the task to identify and solve problems of the machine.

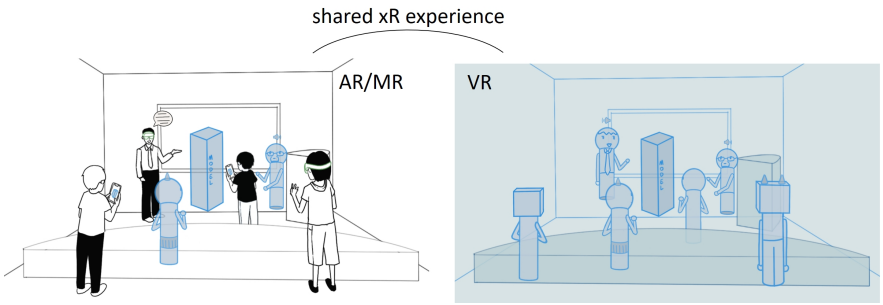

Fig. 1. Users join a university lecture co-located (left) and remotely (right) at the same time. Participants can interact with each other and with the lecturer, all xR content is synchronized and as interactive as necessary. *(pictures by Alistud)*

Figure 1 gives an example for use case #1. Independent of the actual application, users face the following questions:

- Can I use my own device to join a multi-user xR session?
- Do I have to use specific xR hardware or xR software?
- Do I have to use a VR device or an AR device?
- Can I switch between devices if I decide another device might work better?
- **Will my decision on the device limit my experience?**

The last question is particularly very important to answer because it helps find answers for the other questions. If in use case #1 the students pick the

wrong device, they might achieve a bad learning outcome. If in use case #2 the wrong device was chosen, then maybe the interactions do not fit to the task, will take longer, thus result in higher cost for the action. Previous studies have shown interesting differences in usability and user experiences between xR platforms [3].

It is necessary to understand the strengths and weaknesses of each xR device platform, of their interplay and how to support the users in their device decision. Decision makers, educators and users can create valuable concepts and procedures for such scenarios only if solid answers are available to the questions mentioned above. Here, for this paper we call these scenarios **multi-user multi-platform xR** or **cross-plattform xR** experiences.

Fig. 2. Use of four xR platforms. Top left: AR-smartphone. Top right: VR Cardboard. Bottom left: Microsoft HoloLens 2. Bottom right: VR-PC with HP Reverb (G1) and VR controllers [3]

To better understand the named platforms we give the following example devices and configurations as shown in figure 2:

- *PC-VR*: Intel i7 processor equipped with nVidia RTX videocard, HP Reverb head mounted display with controllers.
- *standalone VR headset*: Oculus Quest or Vive Focus with its controllers or hand tracking

- *cardboard VR*: a stock Samsung Galaxy S10 smartphone put in a $5 cardboard with lenses
- *HoloLens 2*: the stock device as it is sold by Microsoft
- *Smartphone based AR*: a stock Samsung Galaxy S10

Table 1 presents major differences between these xR platforms. Most of the platforms can be used for 6 degrees of freedom experiences to include rotational and positional tracking, if correctly implemented.

Table 1. Overview of differences between xR device platforms

Platform	**Computational Performance**	**Interaction**	**Hand usage**
PC-VR	high	controllers or hand-tracked	hold controllers or hands-free for interaction
standalone VR headsets	medium or low	controllers or hand-tracked	hold controllers or hands-free for interaction
Cardboard-VR (smartphone)	medium or low	none or single button	hold device and push button
HoloLens 2	medium	hand-tracked	hands-free for interaction
Smartphone based AR	medium or low	tap or swipe on screen	hold device and touch screen

The computational performance is very different. A common measure is the number of polygons of 3D models that the devices are able to display at sufficient frames per second. For PC-VR this can range from 10 million to more than 100 million. For mobile devices like smartphones and standalone headsets, this is in a region between 100,000 to one or two millions.

The possible interactions are also very different and infer specific use of hands. For PC-VR and standalone VR headsets, hands usually need to hold and use controllers. That means, the controller is used as a medium or metaphor to achieve user's actual intended interaction. Some devices allow hands-free interaction without the need of controllers, such as the Vive Focus 3, Oculus Quest 2 or Microsoft HoloLens 2. Sometimes hands-free interaction can be achieved using extra devices like the LeapMotion. This allows a direct hand interaction with virtual content. With the VR cardboard, users must hold the device at least with one hand, usually with two, and press the single button that is available on some devices. For the smartphone, it is necessary to hold with at least one hand and use the other hand to interact on the touch screen.

In order to investigate user experiences in examples like #1 or #2, a cross-platform xR software framework must be used. This allows one shared xR experience for users of smartphones, AR/MR glasses and VR-PCs. The aim is to use such a framework to study differences between the platforms in multi-user sessions to be able to research for better quality of shared xR experiences.

2 Implementation of a Multi-User Multi-Platform xR-Framework

2.1 Requirements

In order to evaluate properties of multi-user multi-platform xR experiences, systems used must allow multiple users to interact in the same scene with common xR devices in real time. The capabilities of each platform are different (table 1). Rendering and computing power of the PC platform is the strongest. Up to a certain degree of data complexity, it is feasible to run the same virtual scene on different platforms. Our target smartphone is Google Pixel 3. For PC-VR, we selected the AltspaceVR recommended configuration [1] for comparison. As a head-worn device, we selected the Microsoft HoloLens 2 because it is the current industry standard. The Ubi-Interact framework[2] aims at distributed and reactive applications, but currently seems too complex for our purpose.

Table 2. MRTK v2.6.2 supported devices

Platform	**Supported Devices**
OpenXR (Unity 2020.3.8+)	Microsoft HoloLens 2 Windows Mixed Reality headsets
Windows Mixed Reality	Microsoft HoloLens Microsoft HoloLens 2 Windows Mixed Reality headsets
Oculus (Unity 2019.3 or newer)	Oculus Quest
OpenVR	Windows Mixed Reality headsets HTC Vive Oculus Rift
Ultraleap Hand Tracking Mobile	Ultraleap Leap Motion controller iOS and Android

From the xR developer's perspective, each platform must use its own SDK. The PC-VR platform has VRTK[3] (and many more), the Android AR platform has ARCore[4], Apple iOS has ARKit[5] and finally Microsoft HoloLens 2 has MRTK[6]. One specialty about MRTK is, as shown in table 2, that it not only supports HoloLens 2. It also allows to deploy to PC-VR headsets, Android smartphones, iOS smartphones and other platforms. MRTK is able to configure an xR application towards the platform specific requirements. But it does not support collaboration of multiple users on different xR platforms at the same time. Unfortunately, MRTK does not support the Google Cardboard VR platform, yet.

[2] https://wiki.tum.de/pages/viewpage.action?pageId=190677154
[3] https://vrtoolkit.readme.io/docs/summary
[4] https://developers.google.com/ar/develop
[5] https://developer.apple.com/documentation/arkit/
[6] https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/

Therefore for this paper, we decided to ignore Cardboard VR development even if it may be a popular device due to its low price.

Table 3. Data that must be shared between xR devices

data type	**data source**	**update requirements**
screen/head position	coming from all devices	real-time
gaze direction	coming from all devices	real-time
position and pose of controllers	coming from VR	real-time
position and pose of hands and fingers	coming from HoloLens 2	real-time
origin of world coordinate system	registered between all devices	once per start
position and pose of spawned 3D objects	coming from all devices	real-time (interative spawn object) / once (static objects)
state of variables and function calls	coming from all devices	real-time
3D-Mesh of the "real world" around selected AR users	coming from one selected AR user	not necessary in realtime

Data transmission between users is required to allow for interaction and collaboration in the same scene in real time. Table 3 gives examples of data that must be shared and synchronized among all xR devices, independent of the platform used. There are SDKs on the market to solve the data transmission problem, such as Photon [14] or Mirror [9]. They have powerful functions, provide a variety of data transfer methods and are either closed-source or complex to develop with. None of them allow cross-platform xR collaboration as per default.

One major requirement for all multi-user xR experiences is to work in a common shared coordinate system. Currently, if AR users start their application, the origin of their coordinate systems will differ. That is because, at the moment the application starts, each device builds its own map of spatial references. Therefore, the origin of each device's world coordinate system will be different. If, for example, co-located AR users use different coordinate systems, they will see the AR objects in different real-world positions. This would make it hard or impossible to collaborate. To solve this problem, it is possible to use:

- Cloud anchors [2]
- Fiducial markers such as QR codes [17]
- Additional tracking hardware on the device for referencing, such as active LEDs tracked by external tracking systems
- Use of an on-device referencing procedure that processes own tracking data with tracking data of a master device (see section 2.5).

Finally, besides on-device computation, it would be possible to use cloud-based remote rendering. This is a foreseeable and valuable possibility to allow for high-quality real-time renderings on low-performance remote devices. For such scenarios it is also important to know and implement device-specific interaction metaphors and data sharing methods.

In order to understand differences between the named xR platforms it is necessary to create a suitable research platform that fulfills all previously mentioned requirements. Our aim is not to release such a system as a product. Instead, we want to use the system for experiments and research to help development of future multi-user multi-platform xR frameworks:

1. It should be ready for experiments as soon as possible. (simple architecture)
2. The data should be managed from one device. (simple management)
3. The number of users will be less than 10. (simple network system)
4. The future requirements and usage will remain similar. (static architecture)
5. The maintenance cost should be as low as possible. (simple code)

These are used as guidelines for our development. We are not sure if we shall call the developed tools a "framework", "development platform" or "system". In our eyes, it can be all that.

2.2 Concept

Users want to move around in the scene, interact with objects and observe other users' behavior. As shown in figure 3, users may use various inputs to control their avatars and interact with objects, while getting feedback from the scene.

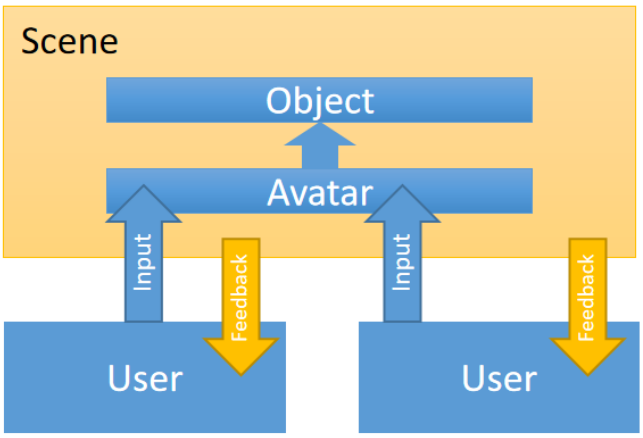

Fig. 3. Block chart for user's xR experience

Table 4. input data of different platforms

Platform	Input data 1	Input data 2	Input data 3
VR-PC	Headset position and rotation	Controller position and rotation	Controller pointed at position
HoloLens 2	Headset position and rotation	Hand and finger position and rotation	Finger pointed at position
AR smartphone	Smartphone position and rotation	screen tap position	Camera focus position and tapped object

The user's input data are different between devices. For example, VR headsets have tracking data from controllers, while smartphones do not. Their input data are summarized in table 4.

If 3D objects are moved in the xR scene, they not only need to feed back the new position and rotation to the user. Instead, they also have to populate that data to other users. Even more, when some functions are triggered by one user, it is necessary to remotely trigger the same functions on other users' systems (remote procedure call, RPC). Ideally, this happens with close-to-zero latency. So for 3D objects in a multi-user scene, there are two kinds of data to be synchronized: First, object's pose data and second, RPC data. There are several basic network architectures to exchange data between devices:

1. Client-server model (C/S): The client directly accesses the server to obtain data, and the server also directly sends data to the client.
2. Three-tier and N-tier models derived from C/S model [15]: As shown in figure 4 (left), they divide different functions into different layers, and each layer performs its own duties to complete the user's request. This architecture is mostly used in complex systems.
3. Peer-To-Peer model (P2P): This is a completely different architecture from the C/S model, as shown in figure 4 (right), P2P emphasizes that no device is dedicated to providing services, but divides the work to each device, and each device acts as a peer. Peers can serve both as clients and as servers [18].

While these models apply to general problems in computer science, Peciva [12] gives a good overview on implications for xR. For example, some of the data needs real-time update (i.e. user's positions) while others don't (i.e. spawning of static geometry).

With regards to the guidelines of the last section, the C/S model and its derived architecture is a good choice [15]. For P2P, handing over the work to each device may affect the performance of less powerful devices (smartphone, HoloLens 2), thus result in a bad user experience. Besides, from the viewpoint of it being an experimental platform, monitoring data flow in the C/S model is much easier than with P2P. Therefore, the C/S model is very suitable for the development of our system.

One major challenge for any collaborative xR system is management of device-dependent coordinate systems: Microsoft Azure Spatial Anchors (Cloud

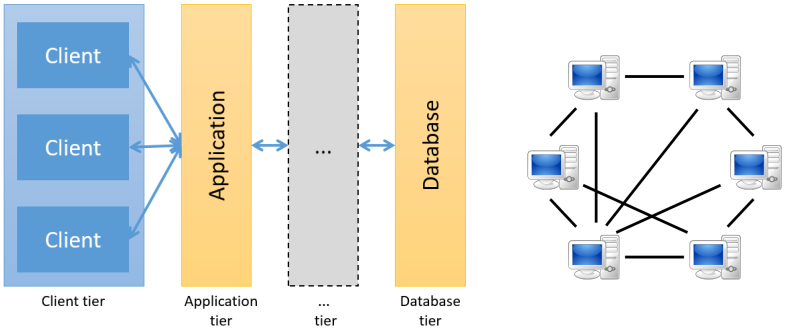

Fig. 4. N-tier model (left) and P2P model (right)

Anchors) require a paid account to log in and upload the anchor to the cloud database, then download it from the cloud to all devices, when in use. This increases maintenance costs and requires an internet connection. As an alternative, a QRcode anchor might be convenient. But it will cause problems in larger rooms when only one QRcode is used for positioning. The third possibility, additional hardware on the device would increase maintenance cost and complexity of the system. As a fourth possible solution, we used a simple calibration method based on 3D-point-set registration (see section 2.5). It only requires the mobile AR user to send at least 4 shared point coordinates to a second mobile AR user to calibrate the coordinate system. A local WiFi network is required to share all data among the devices.

We developed a simple UDP data transmission system: The client connects to the server through the server's IP address and port. Data such as object positions or RPCs can be sent and received after connecting to the server. Data sent to the server will be forwarded to other connected clients. Finally, the conceptual diagram of the system is presented in figure 5.

The whole software is developed with Unity 2019.2 and C# programming language. PC-VR is selected as the VR device, HoloLens 2 as the head-mounted AR device, and an Android mobile phone as the mobile AR device. Because these platforms are all supported by MRTK but Google cardboard is not, Cardboard VR is temporarily abandoned. Because of the cross-platform feature of MRTK, all interactions are implemented with MRTK.

2.3 Acquisition and Transmission of Input Data

The transform component in Unity contains the position coordinates and rotation angle of the object in the scene. In Unity, the main camera represents the user's perspective into the 3D scene. For each platform, MRTK directly synchronizes the device's tracking data to the main camera to achieve the user's xR

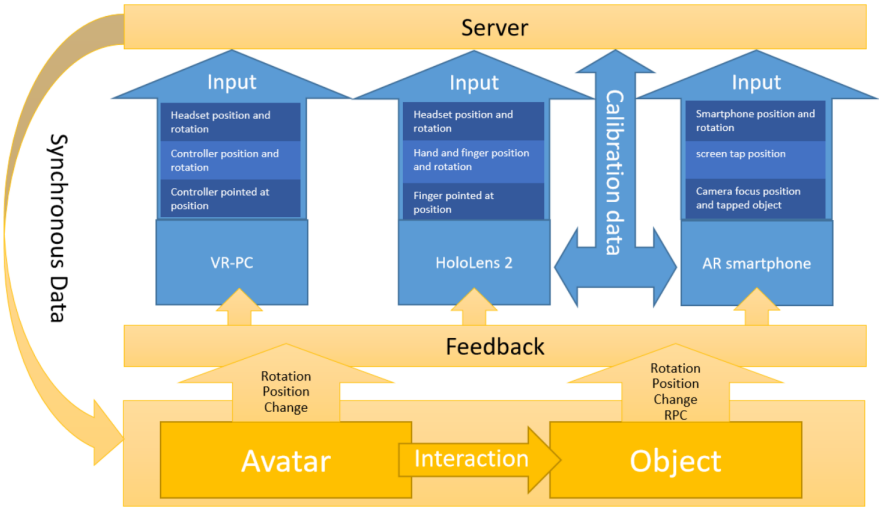

Fig. 5. System concept

experience. Therefore, the main body of avatar can be synchronized by directly obtaining and sending the transform data of main camera to the server.

But for users of PC-VR and HoloLens 2, there is still controller tracking data and hand tracking data to synchronize. As shown in figure 6 (top), both the user's controller and the hand are represented as pointers in MRTK [10].

To obtain the data of a pointer, MRTK's IMixedRealityPointer class is retrieved and saved, which contains the coordinates and rotation angle of the pointer to synchronize (see listing 1.1). For the hand tracking of HoloLens 2, the following listing 1.2 is be used to obtain the transform data of each joint of the palm (see figure 6, bottom).

Listing 1.1. Find all valid pointers

```
var pointers = new HashSet<IMixedRealityPointer>();

foreach (var inputSource in CoreServices.InputSystem.
    DetectedInputSources)
{
    foreach (var pointer in inputSource.Pointers)
    {
        if (pointer.IsInteractionEnabled && !pointers.Contains(
            pointer))
        {
            pointers.Add(pointer);
        }
    }
}
```

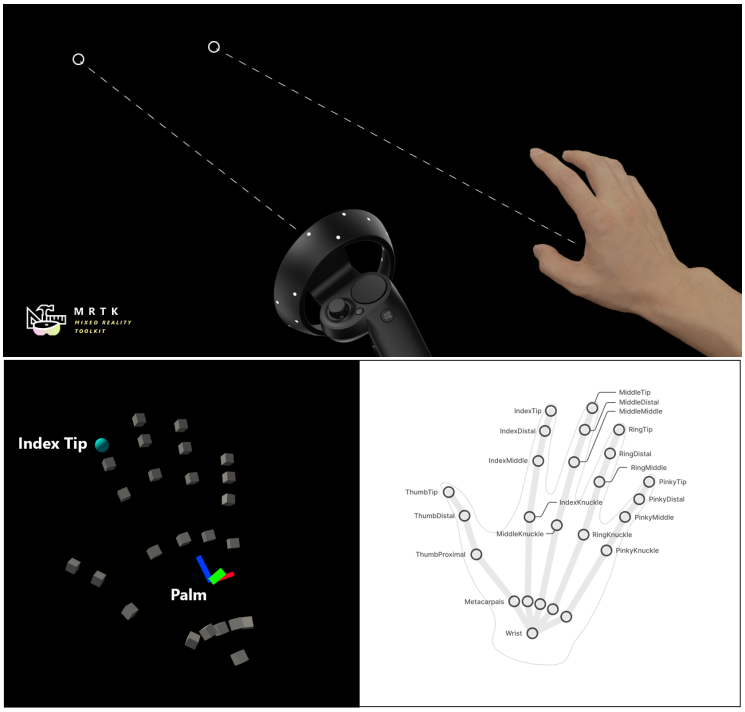

Fig. 6. MRTK Pointer (top) and hand tracking (bottom)

Listing 1.2. Get joint transform from hand joint service

```
var handJointService = CoreServices.GetInputSystemDataProvider<
    IMixedRealityHandJointService>();
if (handJointService != null)
{
    Transform jointTransform = handJointService.
        RequestJointTransform(TrackedHandJoint.IndexTip, Handedness.
        Right);
    // ...
}
```

After obtaining IMixedRealityPointer, the result in the IMixedRealityPointer class directly contains the coordinates of the position pointed at. Because smartphones don't have controllers or hand tracking, the smartphone's pointer defaults to the phone's camera. The focus point is the camera's focus point or where it points after tapping the screen.

The input data acquisition of each platform is summarized in table 5.

Table 5. Input data acquisition of each platform

Platform	Main tracking data	Hand/controller tracking data	Focus point data
PC-VR	Main camera transform	IMixedRealityPointer	IMixedRealityPointer. result
HoloLens 2	Main camera transform	handJointService. RequestJointTransform	IMixedRealityPointer. result
Smartphone	Main camera transform		IMixedRealityPointer. result

2.4 Object Synchronization

The synchronization of the position and rotation angle of the object is the same as the synchronization of the main body of each device, and the data of the transform of the object can be used directly. But how is it possible to synchronize function states? For example, when a button is pressed, the function that makes a light bulb light up is triggered, and the light bulb will light up. How is it possible to trigger the light bulb function on all other user's devices? The concept of "reflection" is used here. When the function is called, the instance's index and MethodInfo are sent to the server, the server forwards it to other users, and other users use methodinfo.Invoke to trigger the corresponding function (listing 1.3). This is what we refer to as RPC here.

Listing 1.3. Sample code for invoking remote procedure calls

```
using System.Reflection;

public void CallMethod(object instance, string methodName)
{
    int index = -1;
    MethodInfo methodToCall;
    for (int i = 0; i < instanceList.Length; i++)
    {
        if (instance.Equals(instanceList[i]))
        {
            index = i;
        }
    }
    if (index > -1)
    {
        Type instanceType = instanceList[index].GetType();
        methodToCall = instanceType.GetMethod(methodName);
        callMethodData.instanceIndex = index;
        callMethodData.methodToCall = methodToCall;
        objectUDPClient.DataEnqueue(callMethodData);
    }
}
```

2.5 Coordinate System Registration

It is necessary to have all co-located AR users use a shared coordinate system. In our case this refers to users of HoloLens 2 and AR smartphone. The 3D-point-set registration method is inspired by the idea of the MPAAM see-through calibration [5]: It requires at least 4 pre-defined spatial input points to generate the required transformation matrix P and thus register the coordinate systems to each other. After P is obtained, any point can be converted between the two coordinate systems A and B:

$$\begin{bmatrix} x_{3D_A} \\ y_{3D_A} \\ z_{3D_A} \\ 1 \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} x_{3D_B} \\ y_{3D_B} \\ z_{3D_B} \\ 1 \end{bmatrix} \tag{1}$$

To obtain P, several steps are necessary. One user must have two AR devices available, that shall be registered to each other. For example, a user carries the HoloLens 2 and an AR smartphone at the same time. Through the HoloLens 2 the user sees the pre-defined target position for the smartphone as a 3D representation. The smartphone and the target position are brought into overlay by moving the smartphone manually. By click of a button, the first 3D-3D correspondence is recorded. This step is done multiple times in different pre-defined positions. The positions must be widely spread to achieve good accuracy.

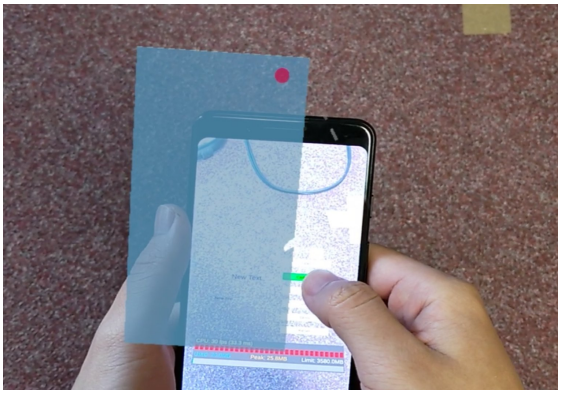

Fig. 7. Image recorded from HoloLens 2 perspective: User trying to overlay the smartphone with the expected pose (blue 3D object) as good as possible.

For each of the 3D-3D correspondences, the data is forwarded to the HoloLens through the server. At the same time, HoloLens 2 also sends the position coordinates of the target positions to the mobile client. HoloLens 2 and the mobile client calculate the calibration matrix through these sets of coordinates. After

all points are recorded the 3D-3D projection matrix P can be calculated. Then, the next participating AR device can follow the same procedure. The expected error rates can be as high as 30 cm, depending on the tracking quality of the involved device's and the accuracy of user's device positioning.

2.6 Framework Test and Discussion

As a first test of the framework, we created an interactive cross-platform simulation using a 3D model of a lathe (see figure 8). The base Unity project was already available at our institution and had to be adjusted to work in the new framework. This was done in less than one work day, without the need to fully understand the existing Unity project.

The following was tested with three users in the scene at the same time, to verify full functionality:

1. Grab and pass objects to each other
2. Operate the levers on the lathe alternately/simultaneously
3. Start drilling functions remotely on the 3D machine
4. Attach / detach objects from the machine by click of a button (RPC)

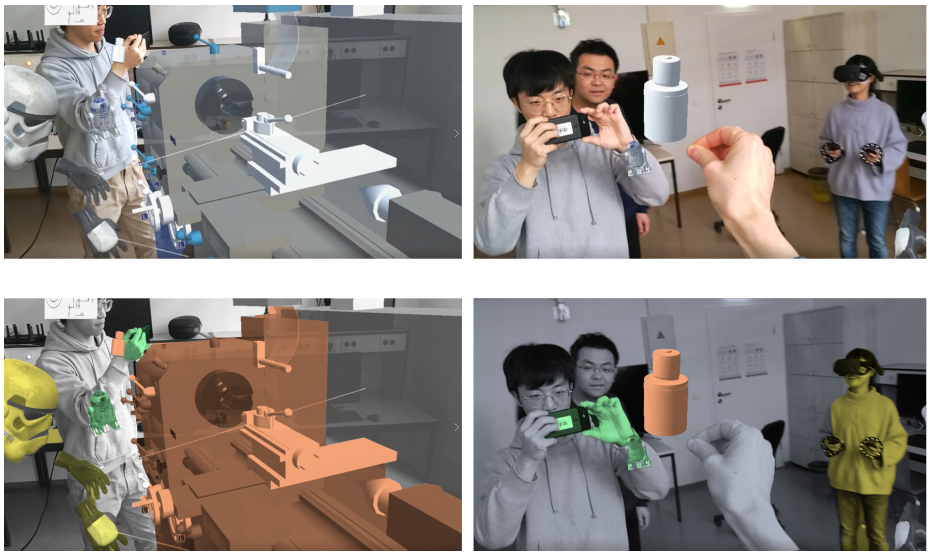

Fig. 8. Top: Screenshots from HoloLens 2 user perspective. Bottom: same images as top, but for better understanding with colored PC-VR user and avatar (yellow), smartphone user and avatar (green), 3D geometry (orange). The registration between smartphone and HoloLens 2 had an error of approximately 20 cm.

A video of the test was recorded and is available online[7]. This confirms that the system can be used for research on interaction and cooperation of multiple cross-platform users in the same scene.

By developing this system, we learned:

1. what data is needed to synchronize avatars and objects, and how to obtain this data
2. Shared coordinate systems are very important for AR users to collaborate.
3. It is possible for users who use different interaction styles to work together.

There are also some questions waiting to be answered. Can users fully understand the behavior of other users? Is it necessary to synchronize all input data? For example, we could only synchronize one or two fingers of the HoloLens 2 user, only sync the controller position of the VR-PC user, or even sync only the focus point. Will these affect the collaboration between users? Following, we conducted preliminary research to start to work on these questions.

3 Evaluation

3.1 Study Goal

The collaborative multi-user multi-platform technology intends to enable collaborative work and social play, implying that we expand from a face-to-face scenario to one in which we collaborate in a virtual world. There is a danger that this new style of work degrades certain critical information for the collaborative process, such as understanding the partner's behavior, intentions and point of view [6].

Therefore, the aim of our first study with the new platform was to evaluate acceptance and usability among different xR device combinations. At the same time it should help us to get a first glimpse of understanding how to work with at least two different xR device types at the same time.

3.2 Method and Procedure

The user study implements three different experiments, always with two users at a time. At the beginning they filled out a demographic survey to assess age and previous xR experience. They were presented the different platforms and could try them out. After that, the experiments begun in randomized order (figure 9). Each participant took part in all three experiments:

1. HoloLens 2 *(user 1)* vs. PC-VR *(user 2)*
2. HoloLens 2 *(user 1)* vs. AR-smartphone *(user 2)*
3. AR-smartphone *(user 1)* vs. PC-VR *(user 2)*

After each run they filled out the user experience questionnaire (UEQ, [16]) and the technology acceptance model (TAM, [11]) questionnaire, both in English language.

[7] `https://www.youtube.com/watch?v=iFT2GO5fosU`

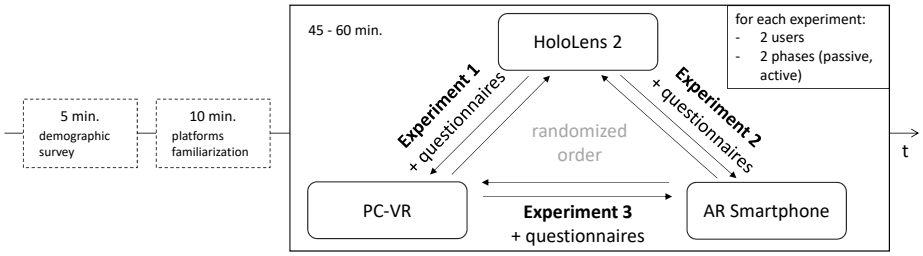

Fig. 9. Procedure of experiment

A puzzle of a 3D hand and arm skeleton had to be solved, similar to our previous work [3]. Eight 3D objects (bones) were spread around and the users had to put them in the right order. There was no need to explain the correct positions because measuring user's task performance was not a goal of the study. For each user there was an active and a passive phase: While *user 1* solved this puzzle with one xR platform, *user 2* passively watched the first user from a different xR platform. Then they changed their role, so that *user 2* had to solve the puzzle while *user 1* watched from afar. During their active and passive phase the participants could talk and help each other to solve the puzzle. They were co-located in the same room. Figure 10 shows a sample setting.

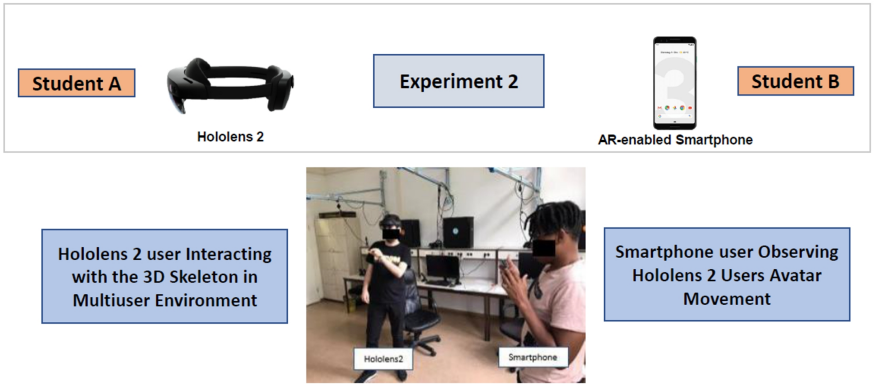

Fig. 10. Sample configuration

The study incorporated $n = 20$ students that participated without any compensation. They were international students of our engineering master degree programs such as Biomedical Engineering or Interactive Media. 30% of the participants were aged between 21-25 years and 70% between 26-30 years, according

to the demographic questionnaire. 60% had prior virtual technology experience. Three had multi-user xR experience from school or virtual reality gaming. The majority of xR experienced participants employed virtual technologies for the purpose of studying. The remaining participants had no prior exposure to xR.

The UEQ captures a broad picture of the user's experience [16]. The user experience (originality and simulation) as well as classical usability (efficiency, perspicuity, and reliability) are evaluated. The participants were required to take a survey consisting of 26 constraint adjective pairs, each on a 7-point scale. The three UEQ measures are attractiveness, pragmatic quality (perspicuity, efficiency, reliability), and hedonic quality (stimulation, originality). Hedonic quality refers to non-task related elements of quality, whereas pragmatic quality refers to work related characteristics. We used the online version of "UEQ Data Analysis Tool" to analyze the data. It transforms data to values between -3 and +3.

Perceived usefulness and perceived ease of use are the two key variables used by the Technology Acceptance Model to determine if a computer system will be accepted by its users. This paradigm has a strong emphasis on the perceptions of potential users. A goal of the study was to investigate a possible link between university students' intentions to utilize two platform software systems and several characteristics such as perceived usefulness, perceived ease of use, attitude, software self-efficacy, subjective norm and system accessibility. Attitude has been identified as a cause of intention. Behavioral intention is a measure on how likely they are to utilize the system. The software self-efficiency is measured by the level of skills required to use this 2-platform multi-user system. Subjective norms as social influencing factors and system accessibility as an organizational factor were only measured by the difficulty of accessing and using this multi-user system in school.

3.3 Results

Matlab was used for statistical analysis. The data was not normally distributed, so we used the Wilcoxon signed rank test with an alpha level of 5%.

In UEQ the values between -0.8 and 0.8 reflect a neutral evaluation of the associated scale. Values larger than 0.8 up to 3.0 represent a positive evaluation, values from -3.0 to -0.8 represent a negative evaluation. From our data we see a positive evaluation in general (figure 11). On the other hand, the scores for novelty are notably lower than others. VR-PC and HoloLens 2 generally give better results in the evaluation compared to AR-smartphone.

From UEQ we did not find any significant differences between the evaluation of the individual platforms from other two platforms and the comparison of the platforms with one another in terms of the superior categories attractiveness, perspicuity, efficiency, stimulation or novelty.

Statistical analysis of the TAM reveals several significant differences:

- Between "HoloLens 2 vs. AR-smartphone" and "AR-smartphone vs. PC-VR" in the subjective norm ($Z = -2.365$, $p = 0.018$).

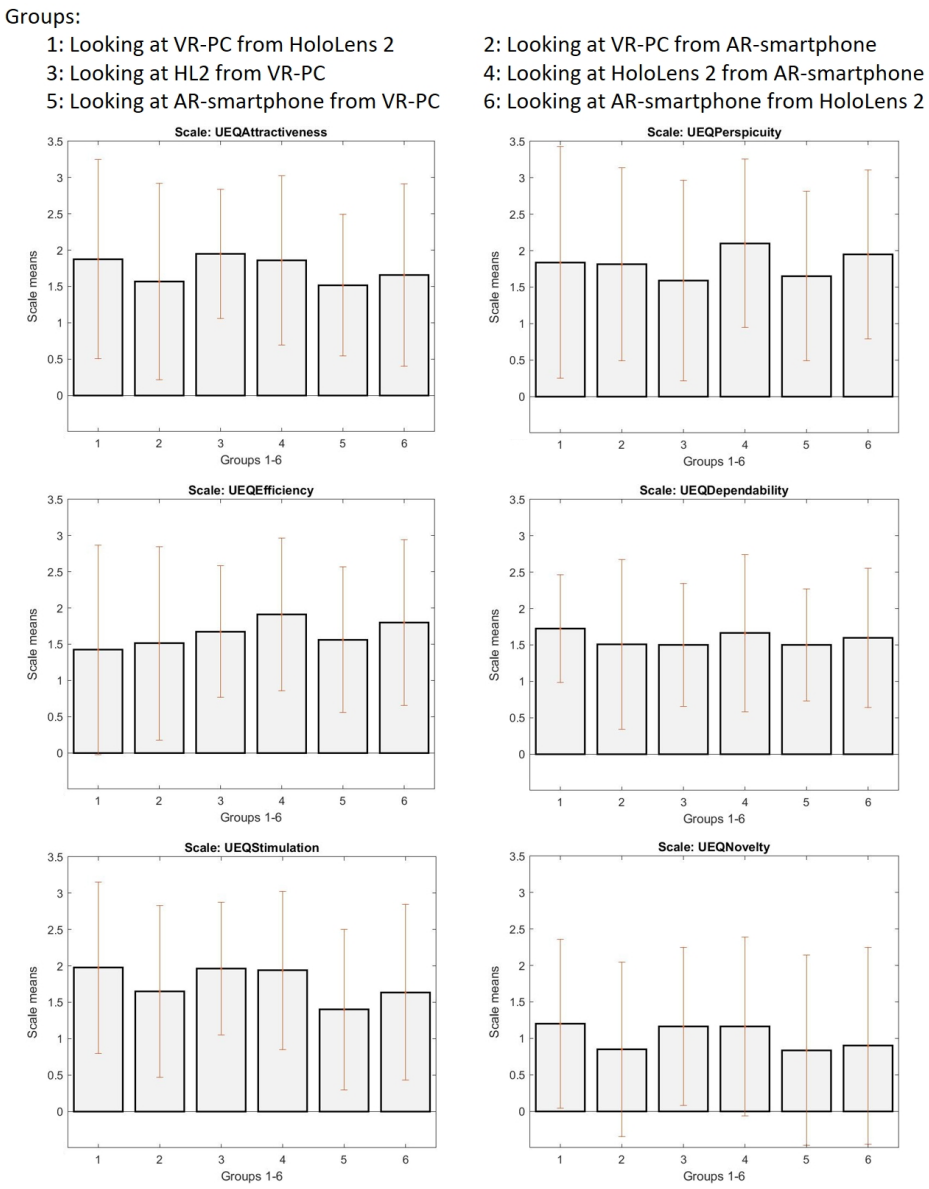

Fig. 11. Mean and standard deviations of the UEQ scales for each group

- "HoloLens 2 vs. PC-VR" compared to "AR-smartphone vs. PC-VR": significant differences for attitude questions 7 and 9 (Z = -2.232, p = 0.026) and (Z = -2.588, p = 0.004).
- "HoloLens 2 vs. PC-VR" compared to "AR-smartphone vs. PC-VR": significant differences for behavioral intention questions 10 and 11 (Z = -2.019, p = 0.043) and (Z = -2.982, p = 0.003)
- "HoloLens 2 vs. PC-VR" compared to "AR-smartphone vs. PC-VR": significant differences for subjective norm questions 13 and 15 with the results of (Z = -2.470, p = 0.013) and (Z = -2.574, p = 0.010), and in software self-efficiency (Z = -1.872, p = 0.061)
- "HoloLens 2 vs. PC-VR" compared to "HoloLens 2 vs. AR-smartphone": significant differences between attitude questions 7 and 8 (Z = -2.153, p = 0.031) and (Z = -2.240, p = 0.025) and in the subjective norm (Z = -2.365, p = 0.018)

3.4 Discussion of Study Results

By use of our novel multi-user multi-platform xR framework, it was possible to conduct a user study with three different xR device types. Users were able to see their partner's avatars and eventually interact with each other - but interaction between users was not part of the experiment.

With the study, we tried to assess how the user's experience, perceived usefulness and perceived ease of use differ if users collaborate through different xR device types. Based on the findings of the TAM questionnaire, we can say that PC-VR in combination with HoloLens 2 is the most suitable multi-user setting, compared to the other combinations tested. This is a combination that would likely be picked by users if they had freedom of choice.

4 Conclusion

It is foreseeable, that with the vision of a "Metaverse" the technological entry barrier must be low, if all people shall be able to access it. It should be possible to use any available xR device to collaborate and experience AR or VR sessions. Until now, there is no public software framework available that reflects the different xR device capabilities and their very different interaction metaphors.
A device-agnostic seamless xR development is not possible yet.

With this paper we present an approach to research on said multi-user multi-platform xR experiences. We developed an exemplary research software framework based on a simple client-server architecture and standard xR SDKs. We were able to use that platform to conduct a first study on xR device combinations from user's perspective. We found that a combination of VR-PC and HoloLens 2 is more likely to be used than other device combinations.

We will continue this work, to understand how to overcome disadvantages of specific xR device types and aim at equal possibilities for all xR users. We hope that in the near future existing xR SDKs such as MRTK will expand their functionality towards simple cross-platform xR collaboration.

References

1. AltspaceVR - minimum system requirements (2022), https://docs.microsoft.com/en-us/windows/mixed-reality/altspace-vr/getting-started/system-requirements
2. Azure spatial anchors documentation - azure spatial anchors (2022), https://docs.microsoft.com/en-us/azure/spatial-anchors/
3. Balani, M.S., Tümler, J.: Usability and user experience of interactions on vr-pc, hololens 2, vr cardboard and ar smartphone in a biomedical application. In: International Conference on Human-Computer Interaction. pp. 275–287. Springer (2021)
4. de Souza Cardoso, L.F., Mariano, F.C.M.Q., Zorzal, E.R.: A survey of industrial augmented reality. Computers & Industrial Engineering **139**, 106159 (2020). https://doi.org/https://doi.org/10.1016/j.cie.2019.106159
5. Grubert, J., Tümler, J., Mecke, R., Schenk, M.: Comparative user study of two see-through calibration methods. VR **10**(269-270), 16 (2010)
6. Hrimech, H., Alem, L., Merienne, F.: How 3d interaction metaphors affect user experience in collaborative virtual environment. Advances in Human-Computer Interaction **2011** (2011)
7. Kavanagh, S., Luxton-Reilly, A., Wuensche, B., Plimmer, B.: A systematic review of virtual reality in education. Themes in Science and Technology Education **10**(2), 85–119 (2017)
8. Martín-Gutiérrez, J., Mora, C.E., Añorbe-Díaz, B., González-Marrero, A.: Virtual technologies trends in education. Eurasia Journal of Mathematics, Science and Technology Education **13**(2), 469–486 (2017)
9. Mirror networking (2022), https://mirror-networking.gitbook.io/docs/
10. Mixed reality toolkit | pointers (2022), https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/input/pointers?view=mrtkunity-2021-05
11. Park, S.Y.: An analysis of the technology acceptance model in understanding university students' behavioral intention to use e-learning. Journal of Educational Technology & Society **12**(3), 150–162 (2009)
12. Pečiva, J.: Active transactions in collaborative virtual environments. Ph.D. thesis, Faculty of Information Technology, Brno University (2007)
13. Perkins Coie LLP: 2020 augmented and virtual reality survey report. Tech. rep., https://xra.org/wp-content/uploads/2020/07/2020-ar-vr-survey-report-0320-v4.pdf (2020)
14. Photon engine | realtime intro (2022), https://doc.photonengine.com/en-us/realtime/current/getting-started/realtime-intro
15. Reese, G.: Database programming with JDBC and Java, chap. Chapter 7: Distributed Application Architecture. O'Reilly (2000)
16. User experience questionnaire (2022), https://www.ueq-online.org/
17. van Schaik, J.: Positioning qr codes in space with hololens 2 - building a 'poor man's vuforia' (2021), https://localjoost.github.io/Positioning-QR-codes-in-space-with-HoloLens-2-building-a-'poor-man's-Vuforia'/
18. Vu, Q.H., Lupu, M., Ooi, B.C.: Peer-to-peer computing: Principles and applications. Springer (2010)