A Low-cost Efficient Approach to Synchronize Real-world and Virtual-world Objects in VR via In-built Cameras

Chaoyi Liu
549602425@qq.com
Xi'an Jiaotong-Liverpool University
Suzhou, China

Rongkai Shi
rongkai.shi19@student.xjtlu.edu.cn
Xi'an Jiaotong-Liverpool University
Suzhou, China

Nan Xiang
nan.xiang@xjtlu.edu.cn
Xi'an Jiaotong-Liverpool University
Suzhou, China

Jieming Ma
jieming.ma@xjtlu.edu.cn
Xi'an Jiaotong-Liverpool University
Suzhou, China

Hai-Ning Liang*
haining.liang@xjtlu.edu.cn
Xi'an Jiaotong-Liverpool University
Suzhou, China

ABSTRACT

Virtual reality (VR) technology has become a growing force in entertainment, education, science, and manufacturing due to the capability of providing users with immersive experiences and natural interaction. Although common input devices such as controllers, gamepads, and trackpads have been integrated into mainstream VR systems for user-content interaction, they cannot provide users with realistic haptic feedback. Some prior work tracks and maps the physical objects into the virtual space to allow users to interact with these objects directly, which improves users' sense of reality in the virtual environment. However, most of them use additional hardware sensors, which inevitably increases the cost. In this research, a lightweight approach is proposed to synchronize the positions and motions between physical and digital objects without any extra costs. We use the real-time captured video data from in-built cameras on a VR headset and employ feature points based algorithms to generate projections of the physical objects in the virtual world. Our approach does not rely on additional sensors but just uses components available in a VR headset. Our approach allows users to interact with target objects with their hands directly without the need for specially designed trackers, markers, and other hardware devices as used in previous work. With our approach, users can get more realistic operational feedback when interacting with corresponding virtual objects.

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; **Mixed / augmented reality**; • **Computing methodologies** → **Tracking**.

KEYWORDS

Virtual Reality, Augmented Reality, Object Synchronization

*corresponding author

VRCAI '22, December 27–29, 2022, Guangzhou, China

ACM ISBN 979-8-4007-0031-6/22/12…$15.00
https://doi.org/10.1145/3574131.3574439

ACM Reference Format:
Chaoyi Liu, Rongkai Shi, Nan Xiang, Jieming Ma, and Hai-Ning Liang. 2022. A Low-cost Efficient Approach to Synchronize Real-world and Virtual-world Objects in VR via In-built Cameras. In *The 18th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI '22), December 27–29, 2022, Guangzhou, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3574131.3574439

1 INTRODUCTION

Virtual reality (VR) is a new practical technology gaining widespread adoption. Currently, VR headsets can simulate a virtual environment and give people an immersive feeling of the environment. In a VR environment, users typically interact with objects in the scene through handheld controllers or a gamepad [LaViola Jr et al. 2017; Nanjappan et al. 2018; Yu et al. 2018, 2021]. For example, when users want to pick up an object in a VR scene, they use a VR controller to select and interact with the virtual object [Lu et al. 2018; Wang et al. 2022; Yu et al. 2021]. While more direct than with traditional computing systems (e.g., using a mouse), it is not as functional and natural compared to the way people interact with objects in real life [Mine et al. 2014]. Such an interaction design reduces the VR immersion. If a physical object (or parts of it) can be projected into the virtual environment, users can interact with it using their hands directly without the need for auxiliary devices, such as controllers, which can improve users' sense of presence and immersion [Kruszyński and van Liere 2009; Lok et al. 2003].

In this work, *synchronization* refers to the process of object localization and mapping between real and virtual worlds in real-time. It offers users a chance to perform interactive tasks in VR without any auxiliary devices like controllers. Existing synchronization approaches suffer from various drawbacks, such as expensive hardware or the inability to track moving objects. With the advancement of VR head-mounted displays (HMDs) technology, VR headsets use inside-out technology to scan the surrounding environment [Gourlay and Held 2017]. The latest VR HMDs are equipped with a multi-angle camera system that can capture their surrounding environment in real-time, making it possible to synchronize the virtual world with the real world through a vision-based method. In this work, we employ a feature detection and matching method to process the image sequence obtained from the VR in-built cameras and estimate the pose of the detected object. To reduce the computational burden and enhance the robustness of the system,

we use simple, non-obtrusive markers to provide stable and reliable features of the target object. After rapid feature matching, our method is able to synchronize the positions of physical and virtual objects and track the moving objects with low-cost hardware in a mixed reality scenario.

2 RELATED WORK

2.1 Synchronizing Real and Virtual Objects in VR

Prior research found that the efficiency of user interaction is highly dependent on the interaction pattern. For example, the same interaction task takes different amounts of time in VR (controller-based interaction) and AR (gesture-based interaction) environments. Krichenbauer et al. [Krichenbauer et al. 2017] designed an experiment to compare the task time consumption between VR and AR. Their user study shows that in a prolonged interaction task, users prefer to interact with tangible objects. Their results demonstrate that synchronizing physical objects in VR allows users to interact with real objects directly, reducing user interaction time and improving the user experience in the long term.

Enhancing immersion by synchronizing real objects with their virtual agents has been an active research field in recent years [Chang et al. 2017], which significantly extends the application scenarios of VR. For example, VR-based simulations have been recognized as a valuable approach to skill training and education. Zhang et al. [Zhang et al. 2019] designed a vision-tangible interactive display method that is based on a binocular optical see-through HMD. It allows users to manipulate a tangible Rubik's cube so as to provide users with a more immersive training experience. They recruited novices who had no experience in using VR as their participants in a user study. Experimental results indicate that VR has a significant advantage in the training of spatial operation tasks compared with traditional methods, and the synchronized vision-tangible interaction method can improve the user experience and can be a promising approach to enhance users' immersive feelings in VR.

A natural environment and haptic feedback can encourage users to exercise or improve spatial cognition. Chang et al. [Chang et al. 2017] combined VR games with spatial awareness training. They built a tangible and embodied interaction scenario using a virtual and real synchronization approach. In their experiments, all the participants expressed a preference for the system, and most of them particularly expressed an appreciation for interacting with physical objects. Tangible VR with the ability to synchronize real and virtual objects has been considered an active research topic with great value and potential.

2.2 Synchronization via Additional Hardware and Sensors

To provide users with suitable tactile feedback in VR, many researchers introduced additional hardware devices to realize the synchronization between virtual and physical objects. Bozgeyikli et al. [Bozgeyikli and Bozgeyikli 2019] implemented a VR game with a tangible spherical sensor (Tangiball). Players in the game were asked to throw a virtual ball at a target. Final points were scored according to where the ball fell on the target. The controller of this game is not a traditional VR gamepad but a tailor-made ball. This spherical controller is a 3D printed transparent plastic ball that has two HTC Vive motion trackers inside. When users throw this ball in the real world, the virtual ball is also tossed into the air in the virtual scene simultaneously. The computer can use data collected from motion trackers to recover the ball's position. Tangiball was considered a relatively simple solution; nevertheless, it expanded the boundaries of the traditional gamepad-based interaction to give users a more realistic interactive experience.

Some researchers went a step further than the previous simple ball-throwing game by introducing more complex hardware systems. For example, Harley et al. [Harley et al. 2017] made a system for tracking tangible objects for VR narratives. The system consists of VR headsets, four different shapes of objects with custom-designed sensor units inside. The four objects ranged from simple to complex: a box, a plush toy, and two types of plastic toys. The sensor unit consisted of an IMU, a Blend Micro, and a battery. The VR HMD and tracking units were assembled into the system with wired connections. In their experiment, in addition to realizing the primary synchronization of objects to the virtual world, users can interact with virtual objects through sensors. For example, their users can knock the real box to trigger some event in the VR environment. Their user study shows that this synchronization system achieved some positive results. However, the wired connection limits users' degrees of freedom in the physical environment, and there is a time lag between the operation in the real world and the response in the virtual environment.

The above solutions are synchronized by using depth sensors. The hardware used in these solutions is costly and difficult to acquire for the average user. Moreover, solutions that use additional sensors have problems with migration. The size of the sensor and the type of data are entirely different in each system. Even for different target objects, sensors are assembled in different ways. The hardware used in prior work is often incompatible with each of the mentioned sensor solutions, and the resulting software solution is also tailored to their specific needs. All these problems make system migration difficult.

Some researchers have used 3D reconstruction and simultaneous localization and mapping technologies to achieve the synchronization between virtual and real objects. Shahram Izadi et al. [Izadi et al. 2011] proposed KinectFusion, which utilized a Kinect depth sensor to carry out a 3D reconstruction of a real scene. The reconstructed models were used as virtual agents that can be interacted with in a virtual environment. Kinect is a lightweight and mobile depth camera that costs slightly less than the hardware devices used in some other prior work. However, there is no recognition ability for fast-moving objects in their implementation. Once the object moves quickly and changes its position, the synchronization will eventually have problems.

The performance of tracking moving objects in real-time was dramatically improved in Geiger et al.'s work by introducing a camera array system [Geiger et al. 2011]. The camera array was used to recognize and scan the moving objects in space. This design allows them to identify and reconstruct objects in real time at 25 frames per second. Although objects can be recognized in real-time, in their final results, the object and the surrounding scene

were not entirely separated, and there were still some parts of the background around the final generated model. The simultaneous use of multiple cameras also affects the cost and portability of their system.

3 OUR METHOD

Our approach for synchronizing real and virtual objects uses the real-time captured video data from in-built cameras of a VR headset and employs feature points-based algorithms to generate projections of the physical objects in the virtual world. Our approach does not rely on additional sensors and uses components available in a VR headset.

To demonstrate the feasibility of this approach, we built the program and tested it in the Unity3D platform with the Vive Cosmos VR toolkit. The Vive Cosmos has a full inside-out tracking mode, with six cameras capturing most of the user's positive information [VIVE 2022]. The Vive Cosmos gives developers access to the in-built cameras, which can offer higher image quality than other inside-out VR tracking systems. The camera parameters are provided by the official development documentation, which simplifies the camera calibration process. Our approach can be divided into three sub-tasks. They are *feature points detection*, *depth calculation*, and *virtual object mapping*. These three parts form the whole process but the implementation of each module is relatively independent, which is convenient for further iterative developments in the future.

3.1 Feature Points Detection

In this work, we use the image sequence from the VR headset's in-built camera as input and do not add any extra hardware or sensors. The first step of our method is to separate the target objects from the input image and obtain essential information to recover their spatial positions.

In our VR scenario, 3D models of the target objects are predefined instead of being real-time constructed. Because real-time construction of the models not only consumes more resources but also has poor model qualities and real-time performance, inaccurate models will result in poor synchronization performance. In addition, the resolution of the in-built camera is still very low, with only 640×480 pixels, which is even worse than most common webcams. Traditional 3D reconstruction and tracking methods require high-quality images to achieve suitable modeling and tracking performance [Koutsoudis et al. 2014]. Such quality data is not available by the VR in-built cameras. But when building a model of a known object, all we need to get is the location of a few key points on the object. Based on the location of these key points, we can project the corresponding virtual object into the virtual space by matching the feature points between the virtual object and the real object. This allows the virtual model to be synchronized with the real target object. The more key points sampled on a single object, the more degrees of freedom can be synchronized. For example, for a pen, only two key points of the nib and the end of the pen are needed to synchronize all interactive actions with it, except scrolling along its roll axis, including spatial translation and rotation at any angle.

We use key points to synchronize the interaction of real objects in the virtual world. The information that needs to be separated in the image is no longer the full object but sparse key points on the object. One of the most common and efficient ways to detect key points is to use color markers. Therefore, we use color markers to provide point features of target objects and use the HSV color model to identify the corresponding color markers.

The commonly used RGB color space is greatly affected by illumination variation. Although it is more consistent with the observation results of human eyes, it is prone to produce deviations in image processing. Based on this reason, we process the marker detection in the HSV color space, which is closer to people's perception experience of color than RGB. It is also often used in image processing and is considered a better choice for color marker detection and tracking objects with a certain color [Stone 2022].

Table 1: HSV ranges of common colors.

	Red		Yellow	Green	Blue	Orange
H_min	0	156	26	35	100	11
H_max	10	180	34	77	124	25
S_min	43		43	43	43	43
S_max	255		255	255	255	255
V_min	46		46	46	46	46
V_max	255		255	255	255	255

The color hue (H) is measured in angles ranging from 0° to 360°. The H angle for red is set to 0°, green is 120°, and blue is 240°. Their complementary colors are yellow for 60°, cyan for 180°, and purple for 300°[Fairchild 2004]. Saturation (S) indicates the degree to which the color approximates the spectral color. The white component of spectral color is 0, and the saturation is the highest. The value ranges from 0% to 100%. A larger value indicates a more saturated color [Fairchild 2013]. Value (V) represents the brightness of the color. For object color, this value is related to the transmittance or reflectance of the object. The value usually ranges from 0% (black) to 100% (white). The HSV value ranges of common colors are shown in Table 1.

The realization process is divided into three steps. The first step is the transformation of the color space, which converts the default BGR color space of each frame into the HSV color space. The image is then masked. This mask contains the HSV range of the target color. The number of masks can be set according to the number of key points that need to be recognized. Multiple colored key points can be processed simultaneously [OpenCV 2022]. A threshold value range is set for each featured color; then, we can obtain all the feature points that meet the conditions in the image and remove the noise points. Finally, the key points of the object in the image plane can be detected. An example result is shown in Figure 1, where a small block that can be seen as a single key point is detected and tracked in a 2D image plane.

3.2 Depth Calculation

After finding the feature points of the object, what needs to be measured is the distance between the object and the VR HMD. Common depth detection methods include binocular vision depth detection and structured light depth detection. Both depth detection methods

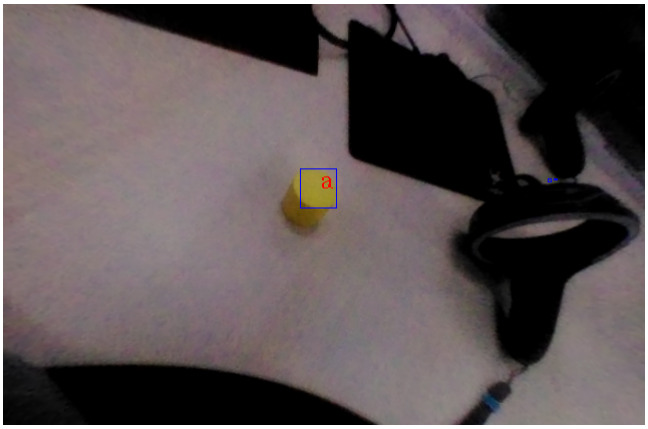

Figure 1: An example result of the HSV tracking recognition.

can be used to calculate within millimeters at relatively close distances. Since depth detection using structured light requires special devices, such as the infrared projector and the infrared camera, which will increase the complexity of the system, the binocular vision method is used for depth detection in our work. The binocular stereo images can be directly acquired from the in-built binocular camera. People have a pair of eyes that can form parallax to an object so that they can clearly perceive the three-dimensional world. Therefore, for a machine, this can be captured via binocular vision, which obtains image information through two cameras and calculates the parallax effect so that the computer can perceive the three-dimensional world. The schematic diagram of a simple binocular stereo vision system is shown in Figure 2.

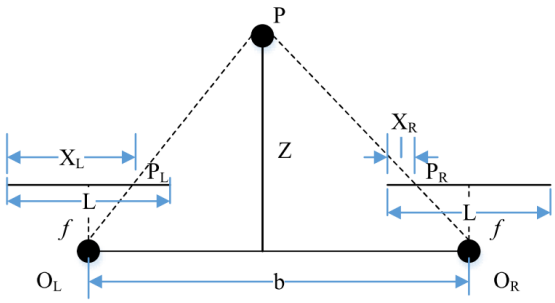

Figure 2: The binocular stereo vision system.

The distance between the projection centers of the two cameras is b, also known as the baseline. Any point P in the three-dimensional (3D) space is P_L at the imaging point of the left camera and P_R at the imaging point of the right camera. According to the principle of straight-line propagation of light, point P in 3D space is the intersection of the line between the projection center point and the imaging point of two cameras. Line segments X_L and X_R are the distances between the left and right cameras imaging points and the left camera imaging plane, respectively, so the parallax of point P on the left and right cameras can be defined as follows:

$$d = |X_L - X_R| \tag{1}$$

At this point, the distance between the imaging points P_L and P_R of the left and right cameras can be expressed as:

$$P_L P_R = b - \left(X_L - \frac{L}{2}\right) - \left(\frac{L}{2} - X_R\right) = b - (X_L - X_R) \tag{2}$$

When the focal length f of the camera is known, the ratio formula is calculated by similar triangle PO_LO_R and PP_LP_R as follows:

$$\frac{b - (X_L - X_R)}{Z - f} = \frac{b}{Z} \tag{3}$$

After simplification, we can get the vertical distance between object P and camera plane O_LO_R:

$$Z = \frac{b * f}{X_L - X_R} \tag{4}$$

In addition to the basic principle of binocular vision ranging, some preparatory work is needed, including the calibration of the binocular camera and the elimination of distortion. The 3D information of objects is derived from 2D images. In order to determine the specific position of the object in the 3D space, besides the information of the image, the parameters of the camera are also needed. In a binocular vision system, apart from the calibration of each camera, the relative positions and transformation matrix between two cameras should be measured in advance. This process is necessary for normal binocular vision implementations, but in our experiment, the development API of the Vive Cosmos provides us with camera intrinsic data after calibrating the HMD.

When the corresponding relationship between the 3D space and the image is determined, in order to calculate the parallax, the corresponding relationship between the points in the three-dimensional space on the left and right images is needed, which is the purpose of stereo matching. Through stereo matching, the corresponding relationship between the points in the left and right images can be defined so as to obtain the parallax and recover the three-dimensional information of the points. In the algorithm, we use the pole line constraint to reduce the complexity of the matching process and improve the speed and accuracy of the stereo matching process. The pole line constraint refers to a point in the left image, and its corresponding matching point on the right image must be on a line, which is the polar line. The polar line constraint can reduce the searching range of images from two-dimensional to one-dimensional, and only need to search on a straight line, which can greatly reduce the complexity of searching and improve the accuracy of matching.

After that, we bring the pixel position of the object obtained in HSV color recognition into the calculation of binocular ranging to get the specific distance of the target point in 3D space. The final result is shown in Figure 3.

3.3 Virtual Object Mapping

After obtaining the position of the key points on the image and the distance estimation of the key points, we can reproduce the key points in virtual space. When the parallax and imaging position are

Figure 3: The final result of the depth calculation.

known, we can obtain the coordinates of corresponding points in 3D space through inversion depth calculation, as shown in Figure 4.

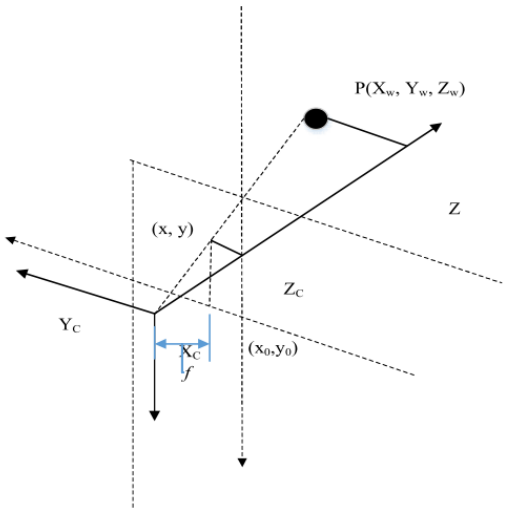

Figure 4: The process to obtain the coordinates of point P in 3D space.

When point P moves in the three-dimensional space, the imaging position of point P on the left and right cameras will change and thus the parallax will also change accordingly. According to the above formula, the parallax is inversely proportional to the distance between the point in the three-dimensional space and the projection center plane. Therefore, as long as we know the parallax of a point, we can know the depth information of that point [Gao et al. 2017]. The image of point P in 3D space is shown in the figure. Based on the similar triangle principle, the relationship can be represented as:

$$\begin{cases} X = \frac{x-x_0}{f} \cdot Z = \frac{x-x_0}{f} \cdot \frac{b*f}{X_L-X_R} \\ Y = \frac{y-y_0}{f} \cdot Z = \frac{y-y_0}{f} \cdot \frac{b*f}{X_L-X_R} \end{cases} \tag{5}$$

The above equations describe how to find the position of a point in 3D space given the position of the image and the corresponding depth, but the process is represented in real space, obtaining real data and then finding the 3D position in virtual space requires some other details. The biggest difference between reading real data and placing it in real space and reading real data and mapping it in virtual space is the conversion of units. We adjust Unity's settings so that units in the real world coordinates and virtual world coordinates reach a ratio of about 1:1. This resolves most of the unit conflicts, but there is one data unit that differs from the others. Those are the coordinates of the object in the picture. The coordinates of an object on a picture are expressed in pixels, but the ratio of pixels to specific units of distance is unknown.

We adjust the parameters of the virtual camera to be the same as the headset cameras in order to solve this problem. In this way, the projection plane in the real world can be synchronized with the projection plane in the VR scene.

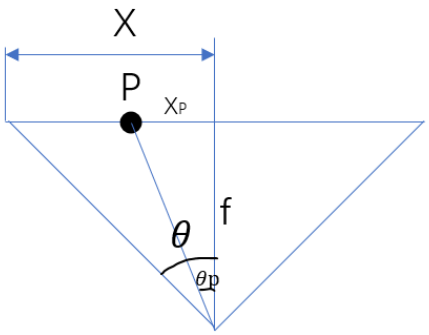

Figure 5: Imaging of the virtual space.

The camera imaging of the virtual space is shown in Figure 5, where X represents half of the picture width in pixels. Since the parameters of the real and virtual cameras are the same, angle θ is 1/2 of the camera's view. So by the properties of a tangent, we can get the following relationship:

$$\frac{\tan(\theta_p)}{\tan(\theta)} = \frac{X_p}{X} \tag{6}$$

So far, we have bypassed the units of X_P to get the corresponding angle θ_P. With θ_P and the depth we get in the depth calculation, the horizontal coordinates of objects in virtual space can be obtained. In the design of the final angle θ, the camera angle is 85°, which is larger than the 60° field of view of common users [Yan et al. 2018]. Before the user approaches the edge of the camera's field of view, he or she can easily adjust the angle of the head to keep the object from falling out of view.

4 EXPERIMENT

4.1 Implementation and Experimental Results

In this section, we demonstrate the experiment design and some results of our method. We present the single-point synchronization

first, then a scenario with multiple key points detection and synchronization. Figure 6 shows the results of the synchronization for a single-point target. The degree of freedom represented by a single point is very limited. However, it clearly illustrates the workflow and effectiveness of our method. This single-point projection can also be used in a simple but common interaction scenario that only requires tracking the 3D translation motion of a single object.

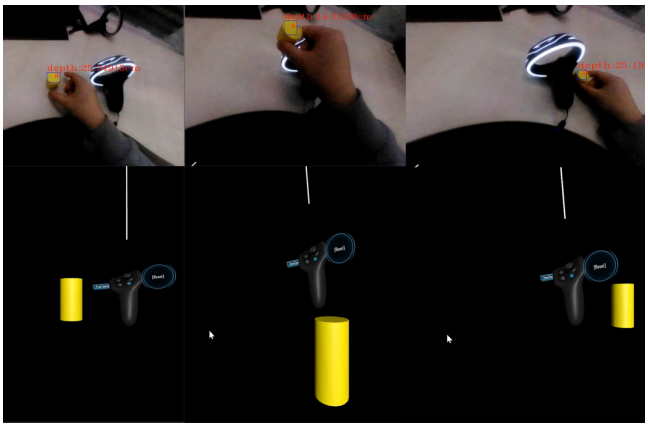

Figure 6: Result of tracking based on a single point.

Because the only spatial change represented by a single point is translation, we use a VR controller as a reference point. As it has a direct connection with the HMD, the VR controller is fully synchronized between the real and virtual worlds, so it is selected as the reference point for the single-point synchronization. The images in the upper row are the scenes that the user can see on the screen, while the images in the lower row are the corresponding VR scenes. The yellow wood block in the image is the target object of single-point projection. In this experiment, the target is a single point, so rotation and other operations cannot be recovered in VR. It can be seen that the yellow block is on the left of the controller at the beginning, which is the same as in the VR scene. The block is then picked up and hovered over the controller. This process is synchronized in the virtual scene. Finally, the block is placed on the right side of the controller. The movements of the virtual block are consistent with the real block.

When there are multiple key points on an object, the object can be synchronized to allow more interaction patterns. Figure 7 shows the simulation of a pen with two key points—the nib and the end of the pen. Through these two points, in addition to spatial translation, the synchronized object can also achieve rotation synchronization, as shown in Figure 7. The upper row images show a rotation process of a pen held in a user's hand. This process is completely synchronized in the VR scene by applying our method as illustrated in the lower-row images.

The two results demonstrate that it is possible to use key points to synchronize real and virtual objects in VR. System latency is restricted to a low-value range that meets a real-time system's requirements. At the same time, the system only uses the in-built camera of the VR headset as the input sensor without introducing other hardware, which is efficient and cost-effective.

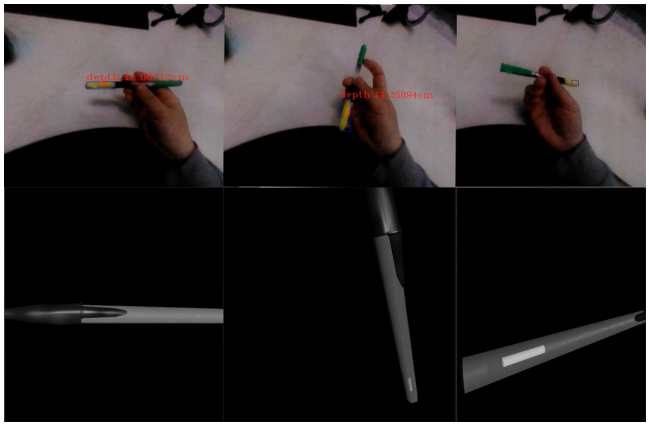

Figure 7: An example of a two-point tracking to simulate a pen.

4.2 Analysis

To analyze the advantages and disadvantages of this synchronization method, we studied several existing and widely used synchronization schemes. The advantages of the proposed scheme can be demonstrated by comparing the existing schemes with our method. The comparison is carried out from three dimensions. They are *hardware cost*, *computing resource usage*, and *time efficiency*.

4.2.1 Hardware Cost. The first is the hardware cost. We investigated the hardware and costs in some prior work. Some of these items are listed in Table 2.

Table 2: Common hardware requirements and costs for tracking and synchronizing objects.

Solutions	Hardware requirements	Cost (USD)
Tangiball [Bozgeyikli and Bozgeyikli 2019]	HTC Vive motion tracker * 2	300
KinectFusion [Izadi et al. 2011]	Kinect	150
VR controllers	HTC Vive controller * 2	300
Camera array	Camera * 10	200
Our method	Color tag(s)	8

In the Tangiball work [Bozgeyikli and Bozgeyikli 2019], Bozgeyikli et al. used two HTC Vive trackers to synchronize the ball. Their experiment connected two sensors to detect the position of the ball. The result is accurate, but the cost is very high due to the use of extra HTV Vive trackers. Their solution requires users to pay about USD 300 as the additional hardware cost without counting other additional materials such as the spherical shell. The most common hardware that uses sensors for synchronization is the VR controller. A pair of Vive Cosmos controllers used in the experiment cost about USD 300 also.

The 3D reconstruction-based implementations also use additional hardware to improve performance. For example, many 3D reconstruction algorithms use the Kinect camera as the input sensor, which costs over USD 150. Similarly, in the real-time 3D reconstruction completed by multiple cameras, a camera array needs at least ten cameras to realize the real-time tracking of objects. Even if the

unit price of the camera in the camera array is as low as USD 20, the total cost is more than USD 200. In contrast to those expensive and cumbersome systems, our method only needs color markers which makes the cost below USD 8.

In addition to the direct hardware costs, our solution is also easier to migrate between different applications. In Table 2, the hardware used by different projects is completely different and so are the target objects that are synchronized with each other. This results in low migration efficiency for the target objects. In the case of a sensor-synchronized sphere, its synchronization approach is difficult to apply to objects of different shapes, such as pens or tables, and chairs. In contrast, our solution simply requires developers to set up markers and implement corresponding algorithms. It can be easily applied to synchronize different objects in various application scenarios, which greatly reduces the system set-up and migration costs.

4.2.2 Computing Resource Usage. The Vive Cosmos API has an AR module that uses 3D localization and mapping algorithms to synchronize real objects into virtual space [SRWorks 2022]. In this section, we compared the computing resource usage of our method with the synchronization algorithm from the Vive Cosmos API.

The two algorithms were tested on a computer with an Intel Core i7-11800H CPU and an Nvidia Geforce 3070 GPU. Computers run at the same voltage to ensure stable CPU and GPU performance. The program is run for 60 seconds in total. The GPU usage of the programs is shown in Figure 8.

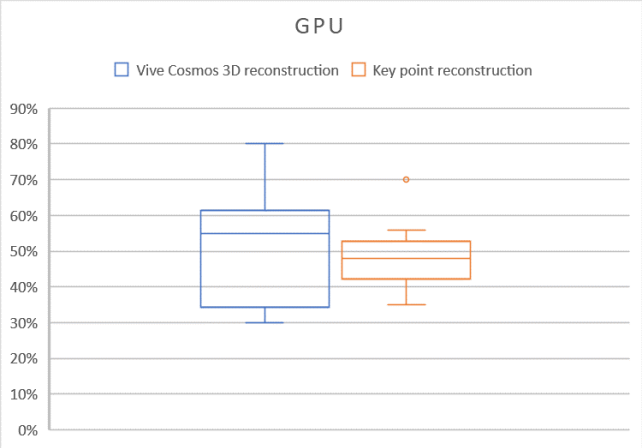

Figure 8: A comparison of the GPU usage between our approach and the Vive Cosmos API.

It can be observed from the boxplot that the average GPU usage of our algorithm is much lower than that of conventional 3D reconstruction and mapping algorithm (around 54%), while the average cost of our algorithm is only about 47%. Furthermore, it can be seen from the distribution of the boxplot that our proposed algorithm is more stable than that of the 3D reconstruction method during runtime. This is because our proposed method is always working in the same state and scanning regardless of whether the object is moving. However, the conventional 3D localization and mapping method is only activated when the object moves.

4.2.3 Time Efficiency. In addition to the consumption of system resources, the time efficiency of the algorithm is also an important performance indicator. The speed of a single scan determines whether the algorithm can synchronize objects in real-time. The test was done by putting the Vive Cosmos 3D reconstruction and mapping algorithm and the key point recognition algorithm into Unity's *Update* method. In Unity, the Update method is executed once per frame, and in synchronous execution, each frame waits for the Update method to be completed before moving to the next frame [Unity 2022]. We count the number of frames per second to get a single run time for each algorithm. The two programs are executed for one minute, and the average frame rate of each program is recorded. The final results are shown in Table 3.

Table 3: Time efficiency comparison.

Solutions	Time Cost	Moving Object
3D Reconstruction	0.5-1.3s per scan	No
Our method	0.05-0.083s per scan	Yes

Our algorithm takes about 0.05 seconds for a single scan, which allows real-time synchronization. The performance of the 3D reconstruction based algorithm is very poor. The fastest single scan is 0.5 seconds and as such, it is hard to track the moving object in real-time. This result also shows that in the same hardware environment, our method has an absolute advantage in running speed.

5 DISCUSSION

In addition to the advantages, we also analyzed some limitations of the proposed method. Binocular vision depth detection faces challenges in some situations. Because the daily use will not always be in an ideal environment, the detection of markers will not always be error-free. In some frames, the recognition algorithm may not find the corresponding marker in the image given by a binocular camera, which would lead to wrong depth estimation. This can occur in environments where the lighting condition is not stable or there are too many interfering colors in the background. Partial occlusion can also have a significant impact on the marker detection process. When the depth of the key point cannot be measured accurately, it can cause tracking drift problems. This type of error has an impact on the final synchronization results, which need to be analyzed and eliminated in the future.

Even so, our method is effective and efficient under normal conditions. In the experiment of pen synchronization, we executed the algorithm with a projected pen for one minute and recorded the number of errors it made. Out of 20 experiments, the average number of errors was 9. In the previous time calculation, the algorithm can scan 20 times per second, which would be about 1,200 scans a minute. So the error rate is only 0.75%. And because errors often occur consecutively, the number of errors perceived by the user is smaller. Such error rates are acceptable in practical use.

In the future, we plan to improve the recognition algorithm of key points and optimize the tracking results to support feature extraction for dynamic or complex scenarios, such as those environments with a faint light or containing objects with similar colors. Another avenue for exploration is about using this approach for controlling

unmanned group vehicles controlled remotely so that objects of interest (e,g, to be avoided [Luo et al. 2022] or to be grasped [Li et al. 2022]) can be tracked dynamically using inexpensive cameras installed on the vehicles. Some filters, such as Kalman filter, which have been shown to be useful in tracking tasks can be explored. With the rapid development of machine learning algorithms, it is possible to further reduce hardware costs and identify errors by updating the traditional color recognition method with efficient machine learning-based approaches.

6 CONCLUSION

In this work, we demonstrated a novel solution for synchronizing virtual and real objects. It allows developers to use the onboard, in-built cameras of a virtual reality head-mounted display (VR HMD) as the input sensor and combines with a feature points detection algorithm and a binocular depth estimation method to achieve object synchronization for real-time interaction in VR. The contribution of this project is mainly twofold. The first is an efficient object synchronization method in VR without adding additional hardware devices. The mainstream solutions for real and virtual synchronization rely on the use of different sensors, whereas our method significantly reduces the hardware cost compared to previous work. The minimal lower cost of hardware makes this new solution more marketable. Second, the detection and mapping algorithms are simplified and optimized. The experimental results have demonstrated that the improvements are straightforward and effective, which not only reduces the use of system resources but also makes it easier to migrate between systems and applications. Overall, our proposed solution can fill the gap in low-cost object synchronization and tracking for VR applications. Compared with high-precision solutions, it has a clear advantage in cost and can also achieve dynamic object tracking.

ACKNOWLEDGMENTS

This work was funded in part by XJTLU Key Special Fund (#KSF-A-03, #KSF-E-65), XJTLU Research Development Fund (#RDF-21-02-065, #RDF-17-02-04), Key Industrial Technology Innovation Fund (#SYG202006, #SYG202122), and Future Network Scientific Research Fund (#FNSRFP-2021-YB-41).

REFERENCES

Lila Bozgeyikli and Evren Bozgeyikli. 2019. Tangiball: Dynamic embodied tangible interaction with a ball in virtual reality. In *Companion Publication of the 2019 on Designing Interactive Systems Conference 2019 Companion*. 135–140.

Jack Shen-Kuen Chang, Georgina Yeboah, Alison Doucette, Paul Clifton, Michael Nitsche, Timothy Welsh, and Ali Mazalek. 2017. Tasc: combining virtual reality with tangible and embodied interactions to support spatial cognition. In *Proceedings of the 2017 Conference on Designing Interactive Systems*. 1239–1251.

Mark Fairchild. 2004. Color appearance models: CIECAM02 and beyond. In *Tutorial Notes, IS&T/SID 12th Color Imaging Conference*.

Mark D Fairchild. 2013. *Color appearance models*. John Wiley & Sons.

Xiang Gao, Tao Zhang, Y Liu, and Q Yan. 2017. 14 lectures on visual SLAM: from theory to practice. *Publishing House of Electronics Industry, Beijing* (2017).

Andreas Geiger, Julius Ziegler, and Christoph Stiller. 2011. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE intelligent vehicles symposium (IV)*. Ieee, 963–968.

Michael J Gourlay and Robert T Held. 2017. Head-Mounted-Display Tracking for Augmented and Virtual Reality. *Information Display* 33, 1 (2017), 6–10.

Daniel Harley, Aneesh P Tarun, Daniel Germinario, and Ali Mazalek. 2017. Tangible vr: Diegetic tangible objects for virtual reality narratives. In *Proceedings of the 2017 Conference on Designing Interactive Systems*. 1253–1263.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 2011. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 559–568.

Anestis Koutsoudis, Blaž Vidmar, George Ioannakis, Fotis Arnaoutoglou, George Pavlidis, and Christodoulos Chamzas. 2014. Multi-image 3D reconstruction data evaluation. *Journal of cultural heritage* 15, 1 (2014), 73–79.

Max Krichenbauer, Goshiro Yamamoto, Takafumi Taketom, Christian Sandor, and Hirokazu Kato. 2017. Augmented reality versus virtual reality for 3d object manipulation. *IEEE transactions on visualization and computer graphics* 24, 2 (2017), 1038–1048.

Krzysztof Jakub Kruszyński and Robert van Liere. 2009. Tangible props for scientific visualization: concept, requirements, application. *Virtual Reality* 13, 4 (2009). https://doi.org/10.1007/s10055-009-0126-1

Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. 2017. *3D user interfaces: theory and practice*. Addison-Wesley Professional.

Ziming Li, Yiming Luo, Jialin Wang, Yushan Pan, Lingyun Yu, and Hai-Ning Liang. 2022. Collaborative Remote Control of Unmanned Ground Vehicles in Virtual Reality. In *2022 International Conference on Interactive Media, Smart Systems and Emerging Technologies (IMET)*. 1–8. https://doi.org/10.1109/IMET54801.2022.9929783

B. Lok, S. Naik, M. Whitton, and F.P. Brooks. 2003. Effects of handling real objects and avatar fidelity on cognitive task performance in virtual environments. In *IEEE Virtual Reality, 2003. Proceedings.* 125–132. https://doi.org/10.1109/VR.2003.1191130

Feiyu Lu, Difeng Yu, Hai-Ning Liang, Wenjun Chen, Konstantinos Papangelis, and Nazlena Mohamad Ali. 2018. Evaluating Engagement Level and Analytical Support of Interactive Visualizations in Virtual Reality Environments. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 143–152. https://doi.org/10.1109/ISMAR.2018.00050

Yiming Luo, Jialin Wang, Rongkai Shi, Hai-Ning Liang, and Shan Luo. 2022. In-Device Feedback in Immersive Head-Mounted Displays for Distance Perception During Teleoperation of Unmanned Ground Vehicles. *IEEE Transactions on Haptics* 15, 1 (2022), 79–84. https://doi.org/10.1109/TOH.2021.3138590

Mark Mine, Arun Yoganandan, and Dane Coffey. 2014. Making VR Work: Building a Real-World Immersive Modeling Application in the Virtual World. In *Proceedings of the 2nd ACM Symposium on Spatial User Interaction* (Honolulu, Hawaii, USA) *(SUI '14)*. Association for Computing Machinery, New York, NY, USA, 80–89. https://doi.org/10.1145/2659766.2659780

Vijayakumar Nanjappan, Hai-Ning Liang, Feiyu Lu, Konstantinos Papangelis, Yong Yue, and Ka Lok Man. 2018. User-elicited dual-hand interactions for manipulating 3D objects in virtual reality environments. *Human-centric Computing and Information Sciences* 8, 1 (2018), pp 1–16. https://doi.org/10.1186/s13673-018-0154-5

OpenCV. 2022. OpenCV: Changing Colorspaces. https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html. (Accessed on 12/06/2022).

SRWorks. 2022. VIVE SRWorks SDK Guide — SRWorks 0.9.3.0 documentation. https://hub.vive.com/storage/srworks/. (Accessed on 12/06/2022).

Rebecca Stone. 2022. Image Segmentation Using Color Spaces in OpenCV + Python – Real Python. https://realpython.com/python-opencv-color-spaces/. (Accessed on 12/06/2022).

Unity. 2022. Unity Scripting API. https://docs.unity3d.com/510/Documentation/ScriptReference/MonoBehaviour.Update.html. (Accessed on 12/06/2022).

VIVE. 2022. VIVE Cosmos | VIVE™. https://www.vive.com/cn/product/vive-cosmos/overview/. (Accessed on 12/06/2022).

Xian Wang, Diego Monteiro, Lik-Hang Lee, Pan Hui, and Hai-Ning Liang. 2022. VibroWeight: Simulating Weight and Center of Gravity Changes of Objects in Virtual Reality for Enhanced Realism. In *2022 IEEE Haptics Symposium (HAPTICS)*. 1–7. https://doi.org/10.1109/HAPTICS52432.2022.9765609

Yukang Yan, Chun Yu, Xiaojuan Ma, Shuai Huang, Hasan Iqbal, and Yuanchun Shi. 2018. Eyes-free target acquisition in interaction space around the body for virtual reality. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–13.

Difeng Yu, Kaixuan Fan, Heng Zhang, Diego Monteiro, Wenge Xu, and Hai-Ning Liang. 2018. PizzaText: Text Entry for Virtual Reality Systems Using Dual Thumbsticks. *IEEE Transactions on Visualization and Computer Graphics* 24, 11 (2018), 2927–2935. https://doi.org/10.1109/TVCG.2018.2868581

Difeng Yu, Xueshi Lu, Rongkai Shi, Hai-Ning Liang, Tilman Dingler, Eduardo Velloso, and Jorge Goncalves. 2021. Gaze-Supported 3D Object Manipulation in Virtual Reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 734, 13 pages. https://doi.org/10.1145/3411764.3445343

Zhenliang Zhang, Yue Li, Jie Guo, Dongdong Weng, Yue Liu, and Yongtian Wang. 2019. Vision-tangible interactive display method for mixed and virtual reality: Toward the human-centered editable reality. *Journal of the Society for Information Display* 27, 2 (2019), 72–84.