applied sciences

MDPI

Article

AnyGesture: Arbitrary One-Handed Gestures for Augmented, Virtual, and Mixed Reality Applications

Alexander Schäfer [1,*], Gerd Reis [2] and Didier Stricker [1,2]

[1] The Augmented Vision Lab, Department Computer Science, TU Kaiserslautern, 67663 Kaiserslautern, Germany; didier.stricker@dfki.de
[2] Augmented Vision Department, German Research Center for Artificial Intelligence (DFKI), 67663 Kaiserslautern, Germany; gerd.reis@dfki.de
* Correspondence: alexander.schaefer@dfki.de

Abstract: Natural user interfaces based on hand gestures are becoming increasingly popular. The need for expensive hardware left a wide range of interaction possibilities that hand tracking enables largely unexplored. Recently, hand tracking has been built into inexpensive and widely available hardware, allowing more and more people access to this technology. This work provides researchers and users with a simple yet effective way to implement various one-handed gestures to enable deeper exploration of gesture-based interactions and interfaces. To this end, this work provides a framework for design, prototyping, testing, and implementation of one-handed gestures. The proposed framework was implemented with two main goals: First, it should be able to recognize any one-handed gesture. Secondly, the design and implementation of gestures should be as simple as performing the gesture and pressing a button to record it. The contribution of this paper is a simple yet unique way to record and recognize static and dynamic one-handed gestures. A static gesture can be captured with a template matching approach, while dynamic gestures use previously captured spatial information. The presented approach was evaluated in a user study with 33 participants and the implementable gestures received high accuracy and user acceptance.

Keywords: gestures; interaction; natural user interface; gestural input; freehand; hands-free; bare hands; virtual reality (VR); augmented reality (AR); mixed reality (MR)

Citation: Schäfer, A.; Reis, G.; Stricker, D. AnyGesture: Arbitrary One-Handed Gestures for Augmented, Virtual, and Mixed Reality Applications. *Appl. Sci.* **2022**, *12*, 1888. https://doi.org/10.3390/app12041888

Academic Editor: Byung-Gyu Kim

Received: 25 January 2022
Accepted: 8 February 2022
Published: 11 February 2022

1. Introduction

Especially for Augmented (AR), Virtual (VR), and Mixed Reality (MR) research and its applications, gesture recognition and hand-gesture-based interfaces are becoming increasingly important. In the earlier years of AR, the interaction was mostly based on physical objects with markers attached to them [1], while more recent applications are using hand tracking for interaction [2–4]. Hand-tracking technology is becoming more reliable and is built into various types of Head-Mounted Displays (HMDs) allowing hand interaction out of the box.

The importance of hand-based interaction is steadily growing, and hand gestures are more frequently used in various application and research scenarios. It is utilized for design and engineering by Vinayak et al. [5] to visualize new concepts and ideas through the use of a hand gesture-driven 3D shape modeling tool for creative expression. Furthermore, hand gestures are also used in vehicles to access various functions while driving without averting one's gaze, as described by Riener et al. [6] and Verma et al. [7]. Medical applications use touchless hand gesture-based interfaces to guarantee safety or sterility during operation [8]. Hand gestures are also used to improve remote collaboration [9,10]. Grasping virtual objects is investigated by Vosinakis et al. [11], locomotion solely based on hand gestures by Schäfer et al. [12], and virtual object manipulation by Song et al. [13]. Recognizing air-writing is introduced by Chen et al. [14] and a character recognition system-based on finger-joint tracking is introduced by Alam et al. [15]. Hand gestures also have an important

role in the the recognition of sign language, as used by Shin et al. [16]. New applications based on hand gestures are constantly being developed. Commonly available hardware to detect and track human body parts contributed significantly to the development of gesture-based interactions and interfaces.

Frameworks from HMD manufacturers usually allow an easy integration of hand gestures for AR and VR applications. These frameworks, however, are often tied to specific hardware and have a limited amount of available gestures. Defining new gestures is not possible or complicated. Additionally, dynamic gestures, such as drawing a sign in the air, concatenation of gestures, or gestures composed of complex hand movements, are not supported.

Researchers implement various gestures with rather complex methodology and require many training samples in order to implement a reliable gesture. One of the most used approaches is the Hidden Markov Model (HMM), which requires many samples to complete graph optimization, and the process of training is relatively complicated. Many existing frameworks are therefore either not generalizable, complex to use, or tied to specific hardware.

This paper proposes a solution with a simple yet effective way to record and recognize any one-handed gesture. The gestures that can be implemented with the proposed framework are ready to be used for many different applications. This contribution has the intention of supporting a variety of hardware and should be seen as a complementary solution and not as a replacement of existing frameworks and toolkits. Part of this work was inspired by the $1 recognizer from Wobbrock et al. [17] in the sense that defining and recognizing gestures should be cheap, easy, and flexible to design. In order to evaluate the proposed methodology, a user study was conducted.

The contributions of this paper are:

- The possibility to capture arbitrary one-handed static and dynamic gestures via button press, enabling rapid design, prototyping, testing, and implementation of one-handed gestures. This includes dynamic gestures with changing hand shape;
- A simple yet unique way to record and recognize dynamic one-handed gestures;
- A comprehensive user study to evaluate the proposed method.

2. Background and Related Work

2.1. Popular SDKs for Hand Gestures

Devices built to support hand tracking usually come with a Software Development Kit (SDK) that provides visualization and simple interactions with virtual objects using hands within immersive virtual environments.

The Mixed Reality Toolkit [18] (MRTK) is a popular choice for AR and VR applications, as there are several sophisticated devices supported. It allows users to manipulate virtual objects by pinching and other rather simple hand gestures. The MRTK relies mostly on virtual objects such as menus, buttons, and sliders for interaction. This SDK should not be considered as a framework for hand gestures but rather a whole toolkit to enable interaction with virtual objects by using natural input, such as hands or eye gaze.

The Leap Motion SDK [19] can be used with specific hardware to enable hand-gesture-based interaction in AR, VR, MR, or even desktop-based systems without HMDs. The SDK provides various options for hand gestures, such as pinching or grabbing. Other gestures can be defined, but the SDK as it stands is limited to static gestures.

Recently, the Oculus portfolio was extended with HMDs that have built-in hand tracking support. The SDK [20] to support this hardware enables hand-based interaction with objects by using simple hand gestures such as pinch, unpinch, and pinch and hold.

In contrast to these SDKs, the framework proposed in this paper allows quickly and easily designing for arbitrary one-handed gestures. Complex dynamic gestures can be recorded and recognized for usage within an interactive system, which is not possible with the aforementioned SDKs.

2.2. Dynamic Gesture Recognition

Several techniques and frameworks for dynamic hand gesture recognition are proposed by researchers [21–27]. Unlike existing solutions, the framework presented in this paper does not require data sets, network training, or expert knowledge to create new gestures.

Studies have been conducted on gesture recognition systems and frameworks in the context of sign language. Galván-Ruiz et al. [28] provide an overview of systems from 1963 to 2020. The discussed papers include a wide variety of different types of devices used to recognize hand signs, e.g., Wi-Fi, RFID, Vision, and Electromyogram data-based recognition systems. Cheok et al. [29] provide an overview of systems between 1995 and 2016 and states that the Hidden Markov Model (HMM) appears to be a promising approach toward dynamic gesture recognition, while Support Vector Machine (SVM) is the most popular method for static gestures. The survey also highlights the accuracy and sample size of the individual systems. For additional background regarding hand gesture recognition systems not necessarily related to sign language, the reader is referred to surveys such as [30–35].

2.3. Gesture Design Tools with Similar Objectives

This section covers the closest gesture-based interface prototyping tools to AnyGesture. These works have similar objectives such as creating gestures without expert knowledge and rapid prototyping of gestures. However, AnyGesture is clearly distinguishable from existing work. A comparison is depicted in Table 1. As an example, Ashbrook et al. introduced the tool MAGIC [36], which allows creating and prototyping gestures with a three-axis accelerometer. Unlike the system presented in this paper, it cannot recognize hand shapes. Speicher et al. [37] proposes GestureWiz, a system that is capable of recording and recognizing gestures from video input data. The system allows rapid prototyping for gestures with a consumer-grade webcam. Compared to this system, AnyGesture works with hand skeleton data and is mainly focused on egocentric vision, which makes it more attractive for AR, VR, and MR applications.

Table 1. Comparison and differentiation with similar systems in existing literature. All mentioned systems have the aim of allowing the rapid design of new gestures. AnyGesture is the only system that does not require multiple training samples and allows gestures that involve individual finger movements. The green check mark indicates availability, the red cross no availability.

	Easy Design of New Gestures	Hand Movement Recognition	Hand Shape Recognition	Individual Finger Movement	No Training Data Required
MAGIC [36]	✓	✓	✗	✗	✗
GestureWiz [37]	✓	✓	✓	✗	✗
Gesture Knitter [38]	✓	✓	✓	✗	✗
AnyGesture	✓	✓	✓	✓	✓

The system Mogeste is proposed by Parnami et al. [39], which allows prototyping and testing of gestures realized with inertial sensors on commodity wearable and mobile devices. Lü and Li [40] propose a system for rapidly prototyping multi-touch gestures. Their system is able to generate gestures that can then be used for multi-touch gesture-based interfaces.

Nebeling et al. proposes Kinect Analysis [41] to inspect and annotate motion recordings obtained from depth cameras that could then potentially be reused to create gestures. Mo et al. introduced Gesture Knitter [38], a tool which allows users to combine fine and gross primitives with a visual scripting language. As an example, users are able to combine a static fist gesture (fine primitive) with a forward movement (gross primitive) to create a "fist-forward" gesture. The authors of [38] train HMMs to recognize gestures that require training samples as opposed to AnyGesture. Furthermore, Gesture Knitter uses the hand's center of mass to track and detect the movements. This allows only static hand shapes to

be formed and recognized, while AnyGesture additionally allows movement of individual fingers to form a gesture. This allows AnyGesture to record and recognize a wider range of gestures, including microgestures where only small subtle finger movements are involved.

3. Materials and Methods

3.1. Possible Gestures

A distinction between two essential types of gestures is made within the proposed framework: static and dynamic gestures (similar to the work of Li et al. [21]):

Static Gestures. A gesture that is detected solely by recognizing a specific hand shape is considered a static gesture. It can be either rotation invariant or tied to a specific hand orientation. Temporal and spatial information, such as the path of the hand, is not considered while recognizing this type of gesture.

Dynamic Gestures. A dynamic gesture uses spatial information, such as the path of the hand or individual joints. This type of gesture is recognized by detecting patterns in movement. Thus, previously defined behavior can be recognized and used. It is to note that stroke gesture recognition, such as that conducted in [17,42,43], is a subset of this category. Dynamic gestures in the proposed system also allow for individual fingers to be moved and recognized as gesture, which is a unique feature compared to similar work in the literature.

A typical static gesture would be a "thumbs up" while drawing a circle in the air with the index finger would be considered a dynamic gesture. Performing a pistol gesture by only moving the thumb down is a dynamic gesture with an individual finger movement.

3.2. Framework Architecture

This section provides a general overview of the implementation of the proposed framework. A crucial component of the framework is the hand data provider, which was implemented to decouple the framework from specific hand tracking hardware and SDKs. It is designed as a wrapper to allow all relevant framework components access to hand tracking data without relying on a specific implementation for hand tracking. In essence, it allows access to hand joints, hand scale, and has support functions to provide data to other components. It generalizes incoming hand tracking data by only relying on joint positions as input. Thus, there is no dependency on specific implementations by manufacturers, and users of the framework only have to make a few adjustments to support a particular device.

The shape of the hand, which is composed of the positions of each individual hand joint, can be stored for the recognition of a gesture. Common hand tracking devices are able to track the Distal, Middle, and Proximal Phalanges of the fingers and provide information about the palm and wrist positions.

While a static gesture is recognized with a previously stored hand shape, dynamic hand gestures need a subset of these points. As an example, a dynamic gesture can be recorded by tracking the position of single or multiple chosen joints. In the process of implementing the framework, it was found that the most important joints are the finger tips, palm, and wrist. Invisible 3D objects are attached to these points in order to allow more interaction possibilities. These invisible objects are basically spheres attached to key positions that can be used to record a spatial path or to perform collision detection (e.g., detect if something was touched by a joint). Collision detection within the hand was found to be particularly useful for some gestures, e.g., the pinch gesture, since a pinch can be reliably detected when the thumb and index finger collide.

To enhance robustness, a subsystem was implemented that observes the movement state of the key positions. It can be considered a post-processing step for the provided hand tracking data. While developing the proposed framework, it was found that such a component is crucial to reliably recognizing gestures since it can prevent many undesired hand gestures. Such undesired hand gestures often occur if too many possible gestures are stored in the system, the user is moving the hands, and no preventive measures are implemented. The position of a hand joint is observed over time, and previously chosen

criteria determine whether a specific joint is currently moving or not. These criteria define the following behavior:

- At which threshold a joint position can be considered still. This is necessary to eliminate noisy data from the hand tracking device.
- How long the joint should be still until it is considered to be not moving.

As mentioned earlier, the motion path of joints is stored. The sampling rate as well as the amount of stored information can be configured. Depending on the desired behavior, a gesture can only be recognized if one or multiple joints are not moving. Each joint is individually observed, but a higher-level component is used to monitor the state of the whole hand.

A rough overview of the main parts that make up the framework is shown in Figure 1. The framework supports an easy swap of the *Hand Tracker* component, allowing many hand tracking solutions to be used with the framework. The *Gesture Recorder* allows recording and saving gestures that are then stored in a set of known gestures. These gestures are then recognized by the *Gesture Recognizer* by comparing stored data with live data from the hand tracker. A *Gesture Interpreter* is used to communicate with the desired application using events that inform when gestures are performed.

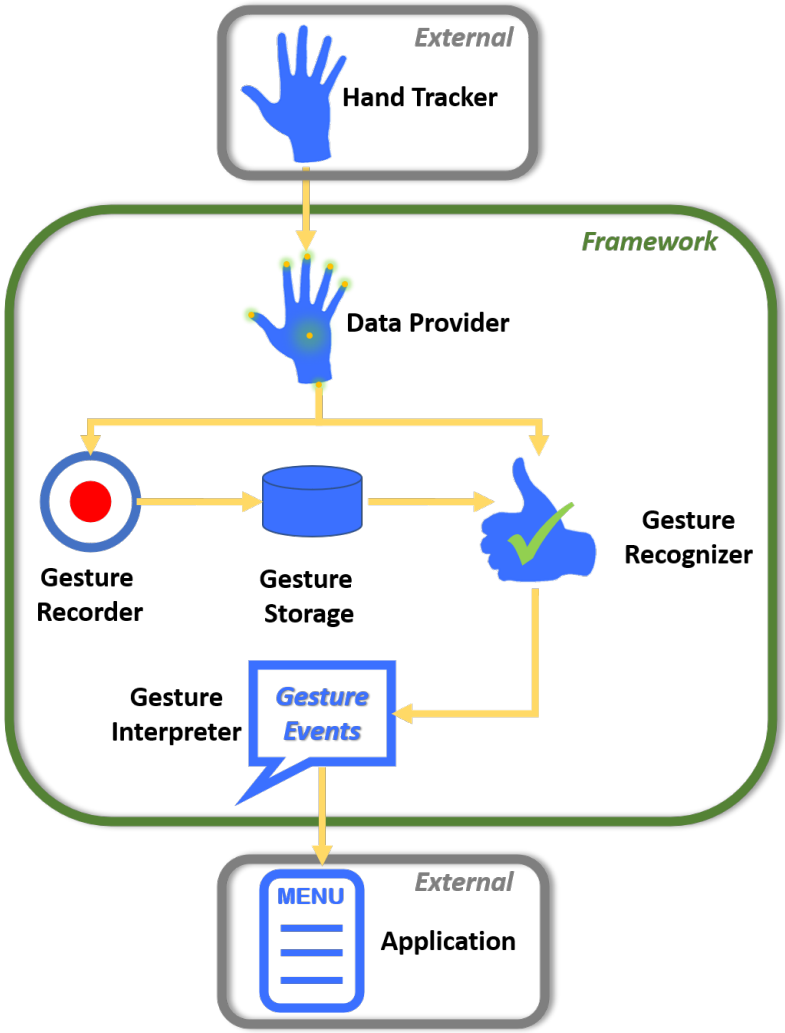

Figure 1. Overview of the proposed framework.

3.3. Tracking

The hand detection in the proposed framework is realized using a hand tracker built into an HMD. The tracking part is crucial and serves as an entry point to the gesture

recognition framework as it is responsible for accurate pose matching. The hand tracker primarily used in the development of the framework provides 23 points for each hand (see Figure 2. Other configurations are supported as well. The more points a tracking device provides, the more fine-grained the gesture recognition is.

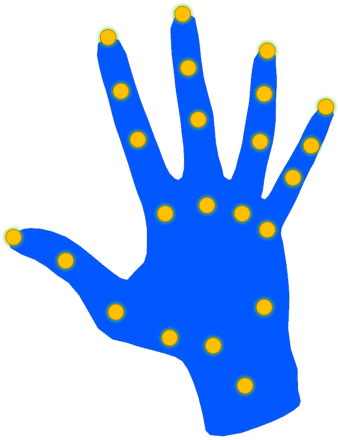

Figure 2. Hand points used to capture static gestures.

3.4. Feature Extraction

The proposed framework requires two primary features for recognizing one-handed gestures:

- **Joint Positions** for static hand shape detection and matching;
- **Finger Tip Positions** for recording spatial information required to perform dynamic gestures.

Hand poses and shapes can be stored for later recognition of static hand gestures, e.g., while a user is performing the hand movement, it can be matched against a predefined set of hand shapes. In order to increase the recognition performance for users with hands below or above the average human hand size, the hand positions are adjusted by a scaling factor. This normalization is necessary to increase the recognition accuracy for gestures that were recorded by a different user than the user performing the gestures. To achieve this, each joint position is divided by the hand scaling factor.

The raw hand shape does not take hand rotation or orientation into account and therefore has rotation invariance. Performing a hand shape that resembles a "thumbs up" will therefore be indistinguishable from a "thumbs down", but it should likely have a different meaning. Furthermore, for some gestures, it is necessary to know whether the hand is facing the user. For example, if a menu should be opened when an open hand gesture in the direction of the user is performed. To solve this problem, the orientation of the hand in relation to the head of the user is utilized. The direction the user is looking at is calculated by using the forward vector of the HMD. Then, the angles between head direction and the X/Y/Z directions perpendicular to the hand at its current position are calculated. This feature can clearly identify the orientation of a hand relative to the VR HMD. This allows for gestures that should be activated when the hand is facing the user or the palm should be upwards, etc. A static gesture is then formed by storing joint positions provided by the hand tracker and the hand orientation relative to the user's facing direction. Static gestures can be captured by pressing a single button on the keyboard. To be more precise, the user shapes their hand as the gesture should be defined and presses a key, and then this shape is added to the set of predefined gestures that can be recognized by the system (see Algorithm 1).

Algorithm 1 Capturing Static Gestures.

```
Desired hand shape is formed by the user
Gesture capture event is triggered via keyboard press
Extract joint positions from current hand pose
Create new gesture object
for Each joint on the hand do
    Transform joint position from world space to local space
    Adjust joint position with hand scaling factor
    Store joint position in gesture object
    Calculate the orientation angles of the hand relative to the head direction of the user
end for
Store calculated hand orientation relative to the users' facing direction in gesture object
Add gesture object to list of known gestures
```

Dynamic gestures, on the other hand, use spatial information for specific points on the hand, e.g., finger tip positions or the palm. The proposed framework allows the recording of spatial information for any joint with arbitrary movements that can be stored for later recognition. This process is depicted in Figure 3. Each dynamic gesture requires a "start" shape, i.e., the system needs to know when the user intends to perform a dynamic gesture. For this reason, each dynamic gesture has a static gesture attached to it, which is later matched to the hand shape from the tracking device during run time. A dynamic gesture can be considered a more complex variant of a static gesture. While a static gesture is performed as soon as it is recognized, it can be used as a cue to activate a dynamic gesture.

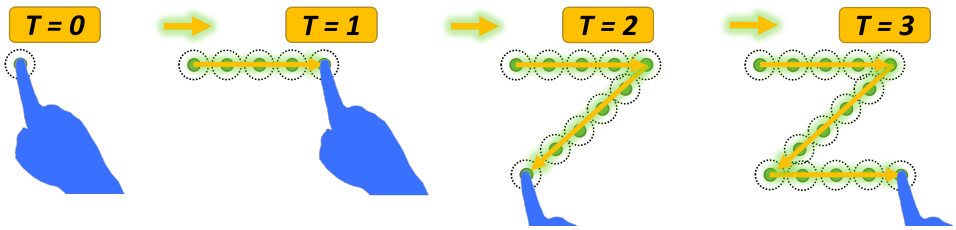

Figure 3. The proposed framework allows recording gestures by storing the spatial path of the desired hand joints as 3D points and represents them as spheres. To help the user redo the desired gesture, a visual path is shown (can be customized or disabled), which the hand joints should follow.

In order to record and store a dynamic gesture, the user shapes their hand as the gesture should be defined and presses a key to start the recording process. The gesture recording component attaches itself to predefined points on the hand, such as the index finger tip, and follows the path of it until the key is pressed again. These predefined points should be chosen by the user before a gesture is being recorded. A gesture that is primarily conducted with the index finger should record the index finger, a swipe gesture might be recorded best if the palm is observed, etc. Which joints should be tracked for which gesture can be decided by the user of the framework. Recording will save a path consisting of 3D points for chosen joints. These 3D points are stored by computing the spatial differences between subsequent points.

The sampling rate of the gesture recording can be freely adjusted (i.e., the number of 3D points stored for a gesture). A high sampling rate means a fine-grained gesture with many points stored, while a lower rate might be more inaccurate but stores less information. It should be noted that a hand tracking device with a low frame rate may cause difficulties in imitating a gesture. This is because hand tracking might introduce stuttering, which cannot reproduce a fine-grained spatial path. Therefore, a low sampling rate is advisable if the target device has a low frame rate. Multiple hand joints can be recorded at the same time. The spatial path is stored as a doubly linked list for each individual joint, i.e., each point knows its predecessor and successor. The list is always initialized with a point at the local coordinates (X,Y,Z) = (0,0,0) set as the first element. This ensures that each

stored gesture path always starts at the initial position of its associated joint during the recognition phase.

3.5. Gesture Recognition

3.5.1. Static Gestures

Static gestures are recognized similar to how they were recorded, e.g., current joint positions are retrieved from the tracking module, hand scaling is applied, and hand orientation is calculated. The values from each hand frame are then compared to the stored values, i.e., gestures. A threshold T_{Shape} can be used to either relax or increase the constraint to detect a gesture. To recognize a gesture, the mean difference for each joint position from the live data and the positions stored inside a gesture are calculated and then compared with the threshold (Equation (1)).

$$|(\sum_{n=1}^{joints} CurrentPos_n - StoredPos_n)| < T_{Shape} \quad (1)$$

The same is conducted with the orientation where OC_x, OC_y, and OC_z represents the current data from the tracker, and OS_x, OS_y, and OS_z are the stored angles between HMD facing and hand directions (Equation (2)).

$$|(OC_x - OC_y - OC_z) - (OS_x - OS_y - OS_z)| < T_{Orientation} \quad (2)$$

The thresholds T_{Shape} and $T_{Orientation}$ can be freely adjusted for each gesture individually within the framework and therefore allow various options for the design of an interactive gesture-based system. For example, a $T_{Orientation} = 360°$ value causes a gesture to be rotation invariant, and the gesture is then detected by shape only, regardless of the hand rotation. It was found that the threshold values $T_{Shape} = 0.05$ and $T_{Orientation} = 15°$ are a good compromise for recognition accuracy of many gestures. A simple algorithm for detecting a single static gesture among a predefined set of static gestures is described in Algorithm 2.

Algorithm 2 Recognizing Static Gestures.

```
New hand frame arrives
Get current hand frame H_current from hand tracker
for Each known gesture in the gesture storage do
    Set D_minimum to Float.Maximum value
    for Each joint on the hand do
        Transform joint position from world space to local space
        Adjust joint position with hand scaling factor
        Calculate distance D between stored joint position and current position
        if D > Threshold then
            Discard current gesture G (hand shape is not matching)
        end if
        Add D to D_Sum
    end for
    if D_Sum < D_minimum then
        Set D_minimum to D_Sum (store the smallest distance)
    end if
end for
if G is not discarded AND D_Sum is the lowest between gestures then
    Current gesture is detected
end if
```

This algorithm is suitable for recognizing the closest matching rotation invariant hand shape in a set of known gestures. In the case of similar gestures inside the gesture

storage, it is a design choice whether the system should take the gesture with the closest matching shape or the closest matching orientation of the hand. The closest matching hand orientation as the final decision to decide for a single gesture out of multiple similar gestures was chosen for the framework. Ultimately, the decision as to which factor (hand shape or orientation) is decisive is rather negligible. This is because the threshold for both factors can be chosen more strictly for each individual gesture. If a gesture is designed more strictly, it will less likely have competing gestures with a similar hand shape.

3.5.2. Dynamic Gestures

In the proposed framework, a dynamic gesture consists of a static gesture and a predefined spatial path for hand joints. A gesture always starts by recognizing a hand shape (the static gesture). After that, the spatial path of the joint positions must be reproduced in order to complete the gesture. A visualization provides the user with information on how to move the hand accordingly. As mentioned in Section 3.4, the spatial path of hand joints, such as finger tip positions, can be recorded, which must be imitated in order to complete a dynamic gesture. This path is reconstructed by placing invisible 3D points in the virtual environment once the attached static gesture is detected. These points are surrounded by a collider that allows them to be touched. These colliders are freely adjustable, which means that a gesture path must be followed quite strictly or the user is allowed to deviate from it. Each 3D point of the path has a specific joint that it must collide with. Recording the path of the index finger will result in 3D points that must be touched by the index finger and not with any other joint, such as thumb, pinky, etc. In order to achieve this, each finger tip on the hand has an object attached to it (invisible to the user), which allows distinguishing which finger has touched which point.

As mentioned before, the points are implemented as a doubly linked list, i.e., each point knows its predecessor and successor. Furthermore, the first point in this list is always set to the world position of the joint it is supposed to be touched with. This will be conducted once the start gesture for the dynamic gesture is found, allowing the dynamic gesture to be performed. After a point is touched, it is marked as such, deactivated, and its successor is activated. If there is no successor left and the current point has been touched, the dynamic gesture was reproduced successfully. This implementation provides a simple yet efficient method for a gesture to be replicated in the correct order. Other constraints are applied in order to ensure that the user imitates the desired gesture correctly. For example, once a dynamic gesture is detected, the first point of the spatial path will be enabled. The system knows which point should be touched next and therefore calculates the distance between the joint it should be touched with and the next point which should be touched. If the distance is too large to the next point, the gesture is considered failed. This test is done for each joint that was used to record the gesture.

3.5.3. Handling Similar Gestures

In a system with arbitrary one-handed gestures, the algorithm to recognize static gestures described in Algorithm 2 is not sufficient since it can only recognize the closest matching gesture. In case a similar hand shape is required for multiple gestures, this algorithm will search for the best match, and therefore, only one gesture will be recognized. In a system with arbitrary hand gestures, however, there could be many similar gestures that should be interpreted differently. The Swipe gesture for example can be performed by swiping left or right with the same hand shape. However, swiping left and right is performed for different desired actions. In the proposed framework, each swipe with a different motion direction would be considered as a separate dynamic gesture, e.g., “Swipe Left” and “Swipe Right” would be recorded separately.

To achieve this, the algorithm described in Section 3.5.1 is adjusted to allow multiple gestures with the same hand shape to be activated at the same time:

- Instead of choosing the gesture with the lowest joint distance (closest match to the stored gestures), all gestures are marked as found that fit within the chosen thresholds.

- If there is more than one gesture found, check if there are gestures marked "Allow Similar Gestures". If yes, then allow all gestures marked as such to be performed. If visualization is enabled, the user receives visual cues for the different motion directions the hand joints can follow.
- If there is more than one gesture found, but no gesture is marked as "Allow Similar Gestures", then only allow the closest match to be performed.

3.5.4. Further Refinement of Dynamic Gestures

To this point, gesture recognition can be refined by comparing stored information with information obtained from current frames. This includes hand shape, orientation, and the recorded dynamic hand path. This is not sufficient for covering arbitrary one-handed gestures. A "pistol" gesture, for example, would require the hand to be still and the thumb to be moved in a specific way, as shown in Figure 4A–C. This can be leveraged by the user by following the expected thumb path with the whole hand instead of moving only the thumb (Figure 4D–F). The framework allows for an option that forces the user to hold the hand still if a specific gesture should be conducted. If a dynamic gesture is detected and this option is enabled, the position of the hand where the gesture was first found is stored. If the distance of the hand is too far from its original point, the gesture will be canceled. This refines the recognition in a way that undesired hand movements no longer fulfill a dynamic gesture path, as depicted in Figure 4D–F. The various refinement options provided by the framework are shown in Table 2.

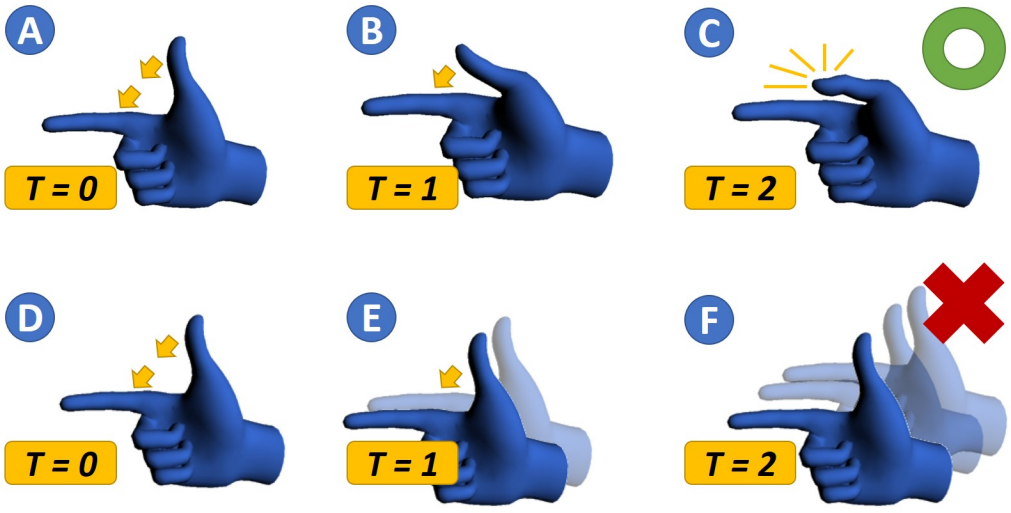

Figure 4. Images (**A**–**C**) show the desired behavior for performing a "pistol" gesture. (**D**–**F**) shows wrong movement that is still accepted by the system if no proper constraints are applied. In this case, the user follows the gesture path by moving the whole hand instead of only the thumb. This can be solved by applying constraints to the palm position of the hand.

3.6. Gesture Interpretation

A gesture in the proposed framework has various events, which enables customizable interpretation. An event can be considered as a callback function that is called when certain conditions are met. These conditions include completing the gesture, canceling the gesture, executing a gesture incorrectly, and gesture progress.

3.6.1. Gesture Completed

This event is called once a gesture is considered performed. For static gestures, this simply means that the current hand frame data matched the stored static hand shape and orientation, and therefore, the gesture was recognized. Dynamic gestures are completed if all constraints attached to the gesture are met. These constraints include following the stored spatial path and others that were mentioned in Section 3.5.4.

Table 2. Various options can be applied to refine gestures in order to implement desired behavior.

Refinement	Effect
Spatial Path	Requires the user to follow a predefined path with specific joints.
Hand Lock	In order to correctly perform the gesture, the user is only allowed to move fingers (see Figure 4).
Allow Similar Gestures	Allows multiple dynamic gestures to begin with the same hand shape.
Pose Threshold	A value that represents how close the current hand shape must match a stored shape.
Orientation Threshold	A value that represents how strict the hand direction should be with respect to the stored values. A high chosen value causes a gesture to be rotation invariant, and it will be detected regardless of the hand rotation.
Visualization	This option enables a visual path that is shown once a dynamic gesture starts in order to help the user perform the gesture correctly.

3.6.2. Gesture Failed

A dynamic gesture can fail due to various reasons, such as not following the spatial path, moving the hand too far away, performing a different static hand shape, etc. Once the system detects that the user violated one of these constraints while a dynamic gesture is active, the framework triggers a callback, which informs other components of the failed gesture and the reason. In the proposed framework, a static gesture cannot fail. This is because it is either being performed or not performed. Once the hand shape and orientation match, a static gesture is already considered complete.

3.6.3. Gesture Progress

Dynamic gestures can notify other components about the percentage of how far the gesture has progressed. This value is coupled to the number of points that were stored during the recording of the gesture. More points in the stored spatial path will result in a finer-grained progress reporting by the gesture. Static gestures do not support progress.

3.6.4. Gesture Activated/Deactivated

Static gestures by default will trigger a Completed event for each hand frame found. If this behavior is not desired, additional framework components can be attached to implement custom behavior. These components will listen to the Gesture Completed event and will trigger an Activation event once. As long as the Gesture Completed event is being triggered, no further Activation event will be sent out by the component. If there was no Completed event for a certain amount of time (can be freely changed and the default value is 0.5 s), a deactivation event is triggered. This is particularly useful to use static gestures to enable and disable certain elements in a virtual environment, such as showing menus or enabling tools.

3.7. Combining Gestures

Gestures can be combined in order to further reflect the desired behavior. As an example, a static gesture can be used to activate and deactivate a hand interaction tool, while a dynamic gesture allows triggering interactions while this tool is available. In Figure 5, a prototypical implementation of a selection tool is shown. The tool is activated by performing a static variant of the "pistol" gesture shown in Figure 4A. Once activated, a visible ray will be shown from the finger tip to the direction the finger is pointing. Enabling this ray also enables the option to perform the dynamic variant of the "pistol" gesture, as shown in Figure 4A–C, which will select an object the ray intersects with.

Besides this example, there are various other implementations possible with the proposed framework to combine and concatenate gestures that allow for truly arbitrary single hand gestures.

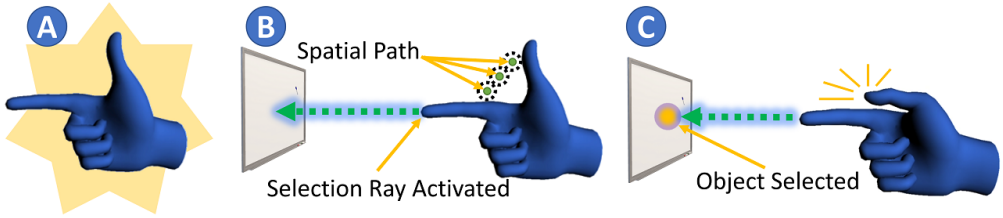

Figure 5. Exemplary implementation of a selection tool using the framework. A recognized static gesture (**A**) activates a visible ray and a dynamic gesture (**B**). If the dynamic gesture is performed by moving the thumbs down, an object is selected that intersects the ray (**C**).

3.8. Choosing Relevant Gestures for the Evaluation

Since the framework supports an arbitrary number of possible gestures, a subset of gestures must be chosen in order to evaluate the framework. It was decided to include a total number of 25 gestures for the evaluation. An overview of all gestures is given in Figure 6. The number of gestures lines up with some existing work that evaluates the recognition of the American Sign Language alphabet, which is usually about 24 distinct gestures. The gestures that denote Z (D3) and J (D4) in the American Sign Language alphabet are usually not evaluated by other works since they are not able to recognize dynamic gestures. It was decided to include a variety of gestures in the evaluation to cover a range of possible use cases. Gestures from the American Sign Language alphabet are included (S1, S5, S15, S16, D3, D4). Some gestures were added due to their complex hand configuration (S3, S12, S16, S17). Furthermore, some gestures were included because they are almost universally understood gestures, such as swiping left and right (D5, D6) and thumbs up and down (S18, S19). Additionally, existing SDKs from hardware manufacturers usually provide a small set of gestures, which were also included in the evaluation, such as an open palm or pointing (S7, D1, D2).

3.9. Choosing a Relevant Assessment

One difficulty in selecting an appropriate assessment is that the existing literature for gesture recognition mostly uses labeled data sets, which allows easily calculating the accuracy of a proposed system within a certain data set. Thus, it can be directly compared to other approaches by using benchmark tests. Since the framework proposed in this article does not use data sets and ground truth data for recognition, calculating the accuracy of gestures within a certain data set is not possible. In a first pilot study, users had to perform a gesture that was shown to them to find out if the approach works as intended. It was observed that users often made a wrong gesture before actually conducting the desired gesture. For example, a participant performed swiping left instead of right or a thumbs up instead of thumbs down. One reason for this is that the participants were eager to finish the test as quickly as possible or were simply inattentive. Hence, a quantitative analysis that can be represented in confusion matrices would be not accurate and therefore not applicable in this case.

For an appropriate assessment of the framework, it is assumed that gestures that are well recognized can also be recognized quickly. Therefore, it was decided to include a task that shows gestures and users should try to perform them as quickly as possible. If a gesture is not well recognized by the framework, it is reflected in the time taken between showing and recognizing the gesture. To further investigate the robustness of the proposed approach, it was decided to include a second part to the experiment. The aim of this task is to investigate how the framework performs when multiple different gestures should be performed in a row. This is interesting for interfaces that require successive gestures to achieve a certain task. For example, a pointing gesture to select something followed by a thumbs up to confirm the selection. For that reason, it was decided to include an additional experiment that involves the user performing multiple successive gestures, namely gesture sets. All gestures from the previous experiment are active in the background (see Figure 6). The number of falsely recognized gestures within a gesture set is recorded. This metric

gives insight how the proposed approach performs when many different gestures should be recognized.

Figure 6. The 25 gestures that were used in the evaluation. S1–S19 are static gestures, and D1–D6 are dynamic gestures.

3.10. Experiment Overview

3.10.1. Objectives

In order to evaluate the framework, a user study is conducted. The reliability of the gestures and whether the gestures can be imitated without difficulty should be investigated. This is especially important since the gestures were recorded and designed from a single user who did not participate in the study. Many different gestures are performed by multiple users. Therefore, it can be investigated if gestures recorded from one user can be reproduced reliably by any other user. Furthermore, it will be assessed how the users perceive the gestures generated by the system. This is a particularly useful insight as end users are often the decisive factor when it comes to selecting a gesture for a task. Subjective questionnaires have also been used in relevant previous work, such as [37,38].

3.10.2. Users

A total of 33 participants were recruited (18 male, 15 female). The participants' ages ranged between 19 and 61 years old ($\mu = 28.6$). The technology affinity of the users was assessed with the Affinity for Technology Interaction (ATI) questionnaire [44,45]. The mean score for the ATI of the participants is 4.30.

3.10.3. Apparatus

The evaluation was performed using a gaming notebook with an Intel Core I7-7820HK, 32 GB DDR4 RAM, Nvidia Geforce GTX 1080 running a 64-bit Windows 10. Hand tracking was realized using the Oculus Quest 2 VR HMD. No controllers were used.

3.10.4. Experimental Task

Participants had to perform gestures in front of a virtual screen while wearing a VR HMD (see Figure 7). After the shown gesture was recognized by the system, a visual confirmation was displayed for one second. Experiment part one required the user to imitate 25 different hand gestures, which were shown one after another. The different gestures are shown in Figure 6. This procedure is repeated five times, i.e., each gesture was performed five times from each participant. The order of gestures was randomized for each repetition and participant. After five repetitions, part two started. For that, ten gesture sets are to be performed by the user. A gesture set is a concatenation of 2–3 gestures that have to be performed in specific order. Gestures that were recognized but were not part of the current gesture set that needed to be performed were recorded as a falsely recognized gesture. (All 25 gestures from experiment part one are enabled in the system during this task.) The gesture sets are described in Appendix A.

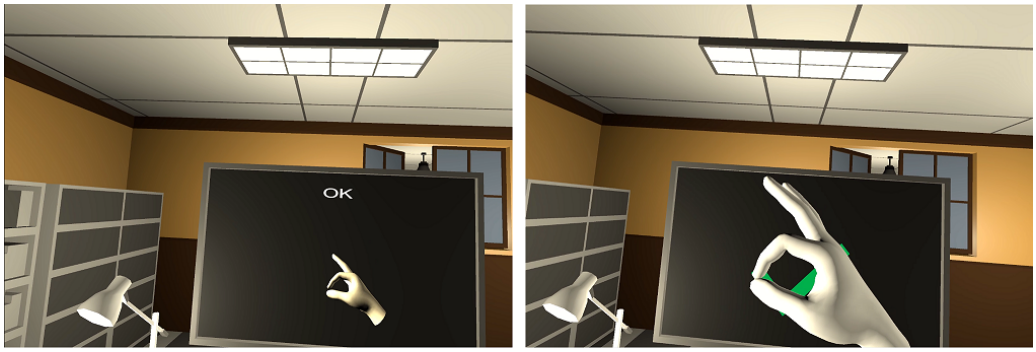

Figure 7. Experiment part one. A gesture is displayed to the participant (**left**) that should then be imitated (**right**). This test is to ensure that gestures can be recognized correctly.

3.10.5. Procedure

Each trial session was conducted individually with the subject. Subjects were naive to the purpose of the experiment. The experimenter explained the trial session procedure, followed by handing out the informed consent. After that, the subjects put on the VR HMD and performed the experimental task. After the task was completed, a questionnaire for the perceived usefulness of the gestures is handed to the participant as well as the ATI questionnaire. The total execution time for one user session was about 30 min.

3.11. Results

A total of 125 gestures from experiment part one and 28 from experiment part two were performed by each participant. With 33 participants, this corresponds to a total number of 5049 gestures performed for the evaluation.

3.11.1. Quantitative Evaluation

Experiment Part One. As a first assessment of whether the gestures produced by the framework are useful, the time required to perform a gesture is measured. The time between displaying the gesture and recognizing it is measured. The mean time to complete the gestures for all participants is depicted in Figure 8. The time until recognition is particularly interesting to find out which gestures were not easy to imitate by users. For example, gesture S17 took longer than most of the gestures, as shown in Figure 8. This might be an indication that gesture recognition configuration was too strict for this particular gesture and could be loosened in order to improve recognition time. A trend that gestures are recognized more quickly with the fifth repetition is noticeable, as depicted in Figure 8. The figure shows graphs for the first, last, and average time for five repetitions. Overall, most static gestures were recognized about one second after being displayed to the user in the fifth repetition.

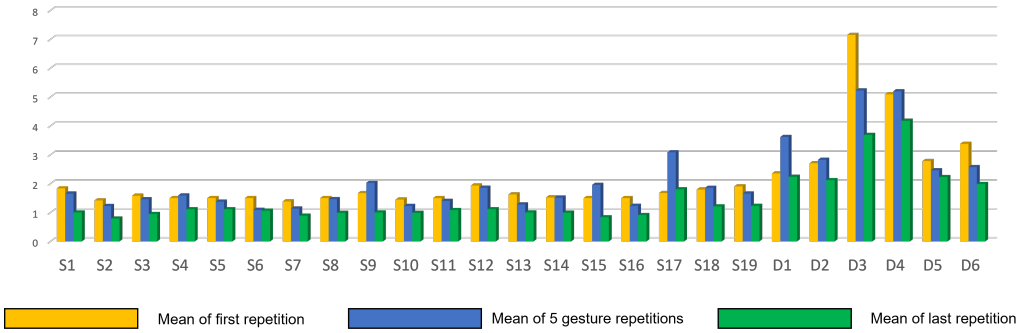

Figure 8. Mean time required by the subjects to perform the individual gestures. The time given refers to the moment when the gesture was recognized by the system after it was displayed to the participant. Mean time for the first time a gesture was performed in yellow, five repetitions of a single gesture is shown in blue, and the last repetition in green.

Experiment Part Two. In the second evaluation phase, the number of times a gesture was falsely recognized is assessed. Participants had to perform gesture sets consisting of two to three gestures that should be performed in a row (see Figure 9). The results are useful in two ways: First, it can be assessed how many falsely recognized gestures are in between each gesture set. Secondly, the framework's capability to produce gestures that can be concatenated. Ten gesture sets are shown consecutively to the user. They are displayed on a virtual screen with visual feedback once a gesture was recognized.

For the ten gesture sets, a total of 28 gestures are to be performed by each participant. This corresponds to a total of 924 performed gestures by participants in this phase. A total number of 51 falsely recognized gestures were recorded during the second phase. This leads to 5.52% of falsely recognized gestures between gesture sets. The results in the second evaluation phase should be treated with caution. During the evaluation, it was found that the test persons often made a wrong gesture because they tried to solve the task as quickly as possible. For example, a "Swipe Left" was often performed until the test persons realized that a "Swipe Right" should actually be performed. Nevertheless, there are some falsely recognized gestures, for example, gesture S1 and S12, because they require forming a similar hand shape. After subtracting the supposedly correctly recognized but incorrectly performed gestures, about 2% of falsely recognized gestures remain. Out of 330 gesture sets performed, seven gesture sets were interrupted with falsely recognized gestures in between.

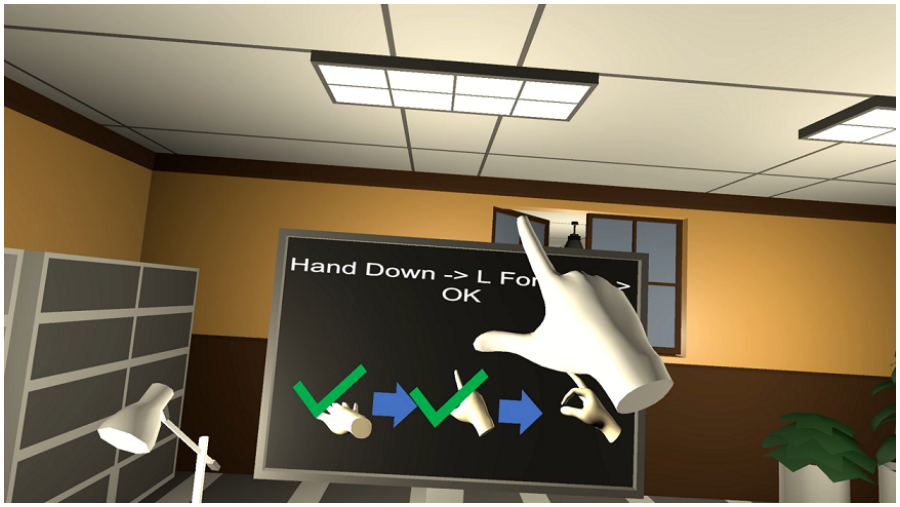

Figure 9. Experiment part two. A gesture set is displayed to the participant to be imitated. The aim is to test whether gestures can be concatenated easily and therefore allow more complex interactions.

3.11.2. Qualitative Evaluation

A questionnaire was handed out to the participants to ask for their subjective opinion of the system. The results of the questionnaire are shown in Figure 10. Three questions were asked:

Q1 The system was able to recognize the gestures I made.
Q2 Dynamic gestures, such as Z or SWIPE, were easy to perform.
Q3 I had the feeling that the gestures I made were immediately recognized.

Participants were overall very satisfied according to the questionnaire. Most users gave either the best possible rating or the second best.

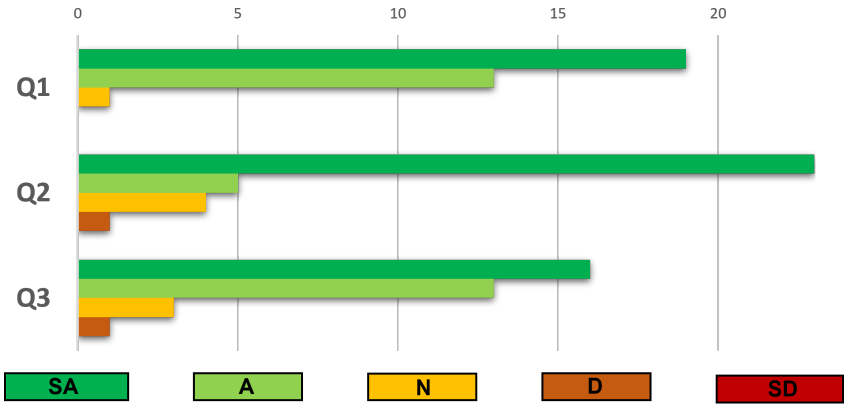

Figure 10. Answers from participants to the given questionnaire. SA = Strongly Agree, A = Agree, N = Neutral, D = Disagree, SD = Strongly Disagree.

4. Discussion

The system implemented with the proposed framework received positive ratings from the participants. The lowest scores in the questionnaire also match with the highest time required to recognize some gestures. For example, a participant gave a neutral rating for Q2 because he had difficulties replicating the Z gesture. Another participant had difficulties performing S9 and gave a neutral rating to Q1. As shown in Figure 8, most static gestures were detected about 1 s after being displayed to the participant. Dynamic gestures require a longer time to complete than static gestures, which is due to the fact that there is hand motion involved. Furthermore, in order for the participant to understand which gesture to perform, a short video is displayed, which needs to be understood first.

Additionally, some recognition times are impacted by the hand configuration itself. For example, some subjects had problems shaping the hand properly due to either congenital characteristics of the hand or previous injuries (especially S3, S10, and S17). Along with the settings that a gesture had to correspond quite closely to the recorded gesture, this resulted in an overall longer recognition time. However, all participants were able to perform the desired gestures, and the questionnaire results are overall very positive. The measurements from the first part of the experiment helped reveal some potential problems with certain gestures. S1, for example, took a surprisingly long time to recognize. The participants had problems turning the hand exactly as it was shown to them in the video.

The proposed implementation shows promising results for recognizing one-handed gestures. To the best of our knowledge, there is no one-handed gesture that cannot be designed, performed, and recognized with the framework. In addition, recording static and dynamic gestures is as simple as performing the desired gesture and subsequently pressing a key. The simplicity of the framework was confirmed when the gestures for the experiment were defined. It took no more than five minutes to design and implement the 25 gestures used in the evaluation. The framework can be used to implement various applications and is not bound to specific hardware. In order to support other hand tracking devices, the hand data provider has to be adjusted to support the target hand tracking SDK.

4.1. Key Takeaways from the Experiment

From the analysis of the results, interviews, and the observation of the participants, the following conclusions can be drawn:

- Hand gestures should be defined in such a way that they can be performed as easily as possible. For gestures S3, S10, and S13, subjects were asked to turn their hand about 180 degrees, which made some subjects uncomfortable. Gesture S3 was also very difficult for one subject to perform because the little finger could not be extended as required for the gesture. When designing gestures, it should be taken into account that people can extend their fingers to different extents.
- The spatial understanding of the test persons differed. The tracing of a letter in the air with the index finger (the Z gesture D3) was sometimes performed incorrectly, which led to frustration for some people. The gesture was aborted if the finger left the path and had to be performed again. This often happened because people underestimated or overestimated the depth of the required path and thus left the gesture path. In general, mistakes should be allowed when users have to follow a path with the finger or a whole hand in order to provide a good user experience with little frustration.
- By observing and interviewing the participants, it can be said that users learned to perform the gestures rather quickly. Looking at the results presented in Figure 8 underlines this argument. Especially within the last repetition, users seemed to be faster at performing gestures. This learning process should be taken into account when evaluating hand-gesture-based interactions. It is therefore recommended to conduct a trial session before the experiment, where users can test and learn the gestures.
- It should always be considered that users perform gestures incorrectly. Regardless of the fact that the experiment was timed, some subjects tried to finish the test as quickly as possible. This led to some gestures being performed incorrectly, such as the thumbs up gesture instead of thumbs down or palm up instead of palm down. This should be taken into account when designing an experiment and recording results to be analyzed.

4.2. Limitations and Future Work

Although both hands can be used to recognize gestures, each hand has its own gesture recognition system attached to it. This means that each hand can perform gestures individually but do not take the other hand into account. Allowing gestures that involve monitoring both hands simultaneously is planned future work. Furthermore, the proposed framework is heavily dependent on the accuracy of the hand tracking data that are fed into the feature extraction component. Highly accurate hand tracking will result in better gesture recording/recognition, while a poorly performing hand tracking device leads to corresponding results. In addition, a gesture recognition system should not only use the hands but other signals of the human body, such as eye gaze, limb movement, and face mimic. Integrating more input modalities is part of future work. A subset of 25 hand gestures was evaluated. The gestures were chosen in order to include some similar and some rather complex hand configurations. Many more gestures are possible with the proposed technique, but the experiments were conducted in order to ensure that the framework works as intended.

As part of future work, it is planned to implement various applications and scenarios that use one-handed gestures. These applications are then evaluated with a focus on the usefulness of specific gestures. Furthermore, automatic detection of similar gestures should be explored in order to eliminate conflicting gestures. Additionally, automatic classification of recorded gestures (whether it is a static or dynamic gesture) would further help gesture designers to record gestures. In the current system, the gesture designer has to manually choose whether to record a static or dynamic gesture. It is also necessary to select which joints of the hand are to be tracked to form a dynamic gesture.

5. Conclusions

In this work, a framework for the design and implementation of one-handed gestures is proposed. It can be used to implement natural user interactions within a wide variety of immersive AR, VR, MR, and desktop applications. Existing hand tracking solutions serve as an entry point to the framework, and the necessary data are redistributed to internal components. These components are responsible for recording, storing, recognizing, and interpreting gestures. The detected gestures can be integrated into applications with events, such as gesture completion, activation, deactivation, failed, and progress.

The goals of the framework are a simple gesture creation process and the recognition of any one-handed gesture. The evaluation confirms that the main objectives of the framework have been achieved. Static and dynamic gestures can be recorded by performing the desired gesture and subsequently pressing a key. For static gestures, hand shape and orientation is stored. Dynamic gestures can be recorded by monitoring joint positions and storing the spatial differences into a doubly linked list. To allow a wide range of complex gestures, several solutions are proposed to enhance and improve gesture recognition. Furthermore, a system to evaluate the framework was built. A user study with 33 participants was conducted, and gestures designed with the framework were evaluated. Overall, the system showed promising results. Gestures designed and implemented by a single person can be reliably reproduced by others without the need for elaborate data sets to define new gestures.

The core of this work is the ease of adding new gestures that are reliably recognized and directly applicable. By not requiring data sets, training, or expert knowledge to implement new gestures, it allows integration into a variety of applications. Even complicated gestures can be easily recorded and tested for usability. The multitude of possible gestures is only limited by the creativity of the users. This work aims to encourage researchers to create gesture-based applications and then examine them for usability, feasibility, and other measures. It paves the way for systems to design, prototype, evaluate, and support arbitrary gestures.

Author Contributions: Conceptualization, A.S. and G.R.; methodology, A.S. and G.R.; software, A.S. and G.R.; validation, A.S. and G.R.; formal analysis, A.S. and G.R.; investigation, A.S. and G.R.; resources, A.S. and G.R.; data curation, A.S. and G.R.; writing—original draft preparation, A.S.; writing—review and editing, A.S. and G.R.; visualization, A.S.; supervision, G.R.; project administration, G.R.; funding acquisition, D.S. and G.R. All authors have read and agreed to the published version of the manuscript.

Funding: Part of this work was funded by the Bundesministerium für Bildung und Forschung (BMBF) in the context of ODPfalz under Grant 03IHS075B. This work was also supported by the EU Research and Innovation program Horizon 2020 (project INFINITY) under the grant agreement ID: 883293.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AR	Augmented Reality
ASL	American Sign Language
HMD	Head Mounted Display
HMM	Hidden Markov Model
MDPI	Multidisciplinary Digital Publishing Institute
MR	Mixed Reality
SDK	Software Development Kit
SVM	Support Vector Machine
VR	Virtual Reality

Appendix A

Table A1. Gesture Sets to be performed by users in evaluation phase 2. The corresponding gestures to the acronyms in the table can be found in Figure 6 of the paper.

Gesture Sets		
Gesture 1	**Gesture 2**	**Gesture 3**
S16	S19	-
D6	S2	S12
S6	S11	S12
D5	S19	-
S8	S14	S13
S11	S16	S8
S13	S18	-
S3	S17	S19
S2	S4	S6
S1	S6	S7

References

1. Billinghurst, M.; Kato, H.; Myojin, S. Advanced interaction techniques for augmented reality applications. In Proceedings of the International Conference on Virtual and Mixed Reality, San Diego, CA, USA, 19–24 July 2009; pp. 13–22.
2. Rani, S.S.; Dhrisya, K.; Ahalyadas, M. Hand gesture control of virtual object in augmented reality. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1500–1505.
3. Sun, Y.; Armengol-Urpi, A.; Kantareddy, S.N.R.; Siegel, J.; Sarma, S. Magichand: Interact with iot devices in augmented reality environment. In Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Osaka, Japan, 23–27 March 2019; pp. 1738–1743.
4. Satriadi, K.A.; Ens, B.; Cordeil, M.; Jenny, B.; Czauderna, T.; Willett, W. Augmented reality map navigation with freehand gestures. In Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Osaka, Japan, 23–27 March 2019; pp. 593–603.
5. Murugappan, S.; Liu, H.; Ramani, K. Shape-It-Up: Hand gesture based creative expression of 3D shapes using intelligent generalized cylinders. *Comput.-Aided Des.* **2013**, *45*, 277–287.
6. Riener, A.; Ferscha, A.; Bachmair, F.; Hagmüller, P.; Lemme, A.; Muttenthaler, D.; Pühringer, D.; Rogner, H.; Tappe, A.; Weger, F. Standardization of the in-car gesture interaction space. In Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, Eindhoven, The Netherlands, 28–30 October 2013; pp. 14–21.
7. Verma, B.; Choudhary, A. Framework for dynamic hand gesture recognition using Grassmann manifold for intelligent vehicles. *IET Intell. Transp. Syst.* **2018**, *12*, 721–729. [CrossRef]
8. Hatscher, B.; Hansen, C. Hand, foot or voice: Alternative input modalities for touchless interaction in the medical domain. In Proceedings of the 20th ACM International Conference on Multimodal Interaction, Boulder, CO, USA, 16–20 October 2018; pp. 145–153.
9. Sasikumar, P.; Gao, L.; Bai, H.; Billinghurst, M. Wearable RemoteFusion: A Mixed Reality Remote Collaboration System with Local Eye Gaze and Remote Hand Gesture Sharing. In Proceedings of the 2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Beijing, China, 10–18 October 2019; pp. 393–394.
10. Schäfer, A.; Reis, G.; Stricker, D. Towards Collaborative Photorealistic VR Meeting Rooms. In Proceedings of the Mensch und Computer 2019, Hamburg, Germany, 8–11 September 2019; pp. 599–603.
11. Vosinakis, S.; Koutsabasis, P. Evaluation of visual feedback techniques for virtual grasping with bare hands using Leap Motion and Oculus Rift. *Virtual Real.* **2018**, *22*, 47–62. [CrossRef]
12. Schäfer, A.; Reis, G.; Stricker, D. Controlling Teleportation-Based Locomotion in Virtual Reality with Hand Gestures: A Comparative Evaluation of Two-Handed and One-Handed Techniques. *Electronics* **2021**, *10*, 715. [CrossRef]
13. Song, P.; Goh, W.B.; Hutama, W.; Fu, C.W.; Liu, X. A handle bar metaphor for virtual object manipulation with mid-air interaction. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Austin, TX, USA, 5–10 May 2012; pp. 1297–1306.
14. Chen, M.; AlRegib, G.; Juang, B.H. Air-Writing Recognition—Part I: Modeling and Recognition of Characters, Words, and Connecting Motions. *IEEE Trans. Hum.-Mach. Syst.* **2016**, *46*, 403–413. [CrossRef]
15. Alam, M.S.; Kwon, K.C.; Kim, N. Implementation of a Character Recognition System Based on Finger-Joint Tracking Using a Depth Camera. *IEEE Trans. Hum.-Mach. Syst.* **2021**, *51*, 229–241. [CrossRef]
16. Shin, J.; Matsuoka, A.; Hasan, M.A.M.; Srizon, A.Y. American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation. *Sensors* **2021**, *21*, 5856. [CrossRef] [PubMed]

17. Wobbrock, J.O.; Wilson, A.D.; Li, Y. Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology—UIST'07, Newport, RI, USA, 7–10 October 2007; Association for Computing Machinery: New York, NY, USA, 2007; pp. 159–168. [CrossRef]
18. Microsoft. Mixed Reality Toolkit (MRTK). 2021. Available online: https://docs.microsoft.com/windows/mixed-reality/mrtk-unity/ (accessed on 25 January 2022).
19. Ultraleap. Leap Motion SDK. 2021. Available online: https://developer.leapmotion.com/ (accessed on 25 January 2022).
20. LLC, F.T. Oculus SDK.2021. Available online: https://developer.oculus.com/develop/ (accessed on 25 January 2022).
21. Li, C.; Zhang, X.; Jin, L. Lpsnet: A novel log path signature feature based hand gesture recognition framework. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Venice, Italy, 22–29 October 2017; pp. 631–639.
22. Zhang, X.; Chen, X.; Li, Y.; Lantz, V.; Wang, K.; Yang, J. A framework for hand gesture recognition based on accelerometer and EMG sensors. *IEEE Trans. Syst. Man Cybern.-Part A Syst. Humans* **2011**, *41*, 1064–1076. [CrossRef]
23. Natarajan, K.; Nguyen, T.H.D.; Mete, M. Hand gesture controlled drones: An open source library. In Proceedings of the 2018 1st International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA, 8–10 April 2018; pp. 168–175.
24. Tang, H.; Liu, H.; Xiao, W.; Sebe, N. Fast and robust dynamic hand gesture recognition via key frames extraction and feature fusion. *Neurocomputing* **2019**, *331*, 424–433. [CrossRef]
25. Liu, L.; Huai, Y. Dynamic hand gesture recognition using LMC for flower and plant interaction. *Int. J. Pattern Recognit. Artif. Intell.* **2019**, *33*, 1950003. [CrossRef]
26. Vaitkevičius, A.; Taroza, M.; Blažauskas, T.; Damaševičius, R.; Maskeliūnas, R.; Woźniak, M. Recognition of American sign language gestures in a virtual reality using leap motion. *Appl. Sci.* **2019**, *9*, 445. [CrossRef]
27. Shao, Q.; Sniffen, A.; Blanchet, J.; Hillis, M.E.; Shi, X.; Haris, T.K.; Liu, J.; Lamberton, J.; Malzkuhn, M.; Quandt, L.C.; et al. Teaching American Sign Language in Mixed Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2020**, *4*, 1–27. [CrossRef]
28. Galván-Ruiz, J.; Travieso-González, C.M.; Tejera-Fettmilch, A.; Pinan-Roescher, A.; Esteban-Hernández, L.; Domínguez-Quintana, L. Perspective and evolution of gesture recognition for sign language: A review. *Sensors* **2020**, *20*, 3571. [CrossRef] [PubMed]
29. Cheok, M.J.; Omar, Z.; Jaward, M.H. A review of hand gesture and sign language recognition techniques. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 131–153. [CrossRef]
30. Chen, L.; Wang, F.; Deng, H.; Ji, K. A survey on hand gesture recognition. In Proceedings of the 2013 International conference on computer sciences and applications, Wuhan, China, 14–15 December 2013; pp. 313–316.
31. Mitra, S.; Acharya, T. Gesture recognition: A survey. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2007**, *37*, 311–324. [CrossRef]
32. Chaudhary, A.; Raheja, J.; Das, K.; Raheja, S. A survey on hand gesture recognition in context of soft computing. In Proceedings of the International Conference on Computer Science and Information Technology, Bangalore, India, 2–4 January 2011; pp. 46–55.
33. Khan, R.Z.; Ibraheem, N.A. Survey on gesture recognition for hand image postures. *Comput. Inf. Sci.* **2012**, *5*, 110. [CrossRef]
34. Sarkar, A.R.; Sanyal, G.; Majumder, S. Hand gesture recognition systems: A survey. *Int. J. Comput. Appl.* **2013**, *71*.
35. Ibraheem, N.A.; Khan, R.Z. Survey on various gesture recognition technologies and techniques. *Int. J. Comput. Appl.* **2012**, *50*, 38–44.
36. Ashbrook, D.; Starner, T. MAGIC: A motion gesture design tool. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Atlanta, GA, USA, 10–15 April 2010; pp. 2159–2168.
37. Speicher, M.; Nebeling, M. Gesturewiz: A human-powered gesture design environment for user interface prototypes. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018; pp. 1–11.
38. Mo, G.B.; Dudley, J.J.; Kristensson, P.O., Gesture Knitter: A Hand Gesture Design Tool for Head-Mounted Mixed Reality Applications. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; Association for Computing Machinery: New York, NY, USA, 2021.
39. Parnami, A.; Gupta, A.; Reyes, G.; Sadana, R.; Li, Y.; Abowd, G. Mogeste: Mobile Tool for in-Situ Motion Gesture Design. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct—UbiComp'16, Heidelberg, Germany, 12–16 September 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 345–348. [CrossRef]
40. Lü, H.; Li, Y., Gesture Coder: A Tool for Programming Multi-Touch Gestures by Demonstration. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Austin, TX, USA, 5–10 May 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 2875–2884.
41. Nebeling, M.; Ott, D.; Norrie, M.C. Kinect Analysis: A System for Recording, Analysing and Sharing Multimodal Interaction Elicitation Studies. In Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems—EICS'15, Duisburg, Germany, 23–26 June 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 142–151. [CrossRef]
42. Vatavu, R.D.; Anthony, L.; Wobbrock, J.O. Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes. In Proceedings of the 14th ACM International Conference on Multimodal Interaction— ICMI'12, Santa Monica, CA, USA, 22–26 October 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 273–280. [CrossRef]

43. Kristensson, P.O.; Zhai, S. SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-Based Computers. In Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology—UIST'04, Santa Fe, NM, USA, 24–27 October 2004; Association for Computing Machinery: New York, NY, USA, 2004; pp. 43–52. [CrossRef]
44. Franke, T.; Attig, C.; Wessel, D. A Personal Resource for Technology Interaction: Development and Validation of the Affinity for Technology Interaction (ATI) Scale. *Int. J. Hum.–Comput. Interact.* **2019**, *35*, 456–467. [CrossRef]
45. Attig, C.; Wessel, D.; Franke, T. Assessing Personality Differences in Human-Technology Interaction: An Overview of Key Self-report Scales to Predict Successful Interaction. In *HCI International 2017—Posters' Extended Abstracts*; Stephanidis, C., Ed.; Springer International Publishing: Cham, Switzerland, 2017; pp. 19–29.