Markus Höll, BSc

Coulomb Friction-Based Realistic Hand-Object Interaction for Augmented and Virtual Reality

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dr. Vincent Lepetit
Institute for Computer Graphics and Vision

Advisor

Dipl.-Ing. Markus Oberweger
Institute of Computer Graphics and Vision

Graz, Austria, Nov. 2017

TO MY PARENTS

UNTIL I MEET YOU AGAIN, DAD

"I seem to have been only like a boy playing on the seashore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me".

- Isaac Newton (1642 - 1726)

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum	Unterschrift

Acknowledgments

First and foremost I would like to thank Univ.-Prof. Dr. Vincent Lepetit. I am grateful that I had the chance to work on such an interesting and complex topic. Honestly, I am thankful for the faith you have had in me. You gave me the chance to aim for high objectives and explore new uncertain paths.

I am particularly grateful for the assistance provided by Dipl.-Ing. Markus Oberweger. It was an amazing experience working with such a professional advisor. I always enjoyed to discuss new potential approaches with you, thank you. Further, I want to express my gratitude to Dr. Christian Pirchheim for being the greatest office colleague of all time. The conversations with you were from high value to me and gave me many new perspectives and impressions. Thanks to you I really enjoyed my time at the Christian Doppler Laboratory (CDL). Also, I want to express my thanks to Dr. Clemens Arth and Dr. Peter Roth from the Institute of Computer Graphics and Vision (ICG).

My special thanks goes to Aleksandra S. who accompanied me through all challenging times during my thesis. Honestly, thank you for your support and comprehension. Last but not least, I want to thank my colleagues, friends and family for their support, encouragement and all the excellent jokes throughout my study.

Thanks for all your encouragement!

Abstract

We propose a physics-based method for dexterous 'real hand'-'virtual object' interaction in Augmented Reality (AR) and Virtual Reality (VR) environments. Our method is based on the Coulomb friction model, and we show how to efficiently implement it in a commodity engine for real-time performance. This model enables unconstrained and very convincing simulations of many types of actions such as pushing, pulling, grasping, or even dexterous manipulations such as spinning objects between fingers without restrictions on the objects' shapes or hand poses. Since it is an analytic model, we do not require any prerecorded data, in contrast to previous methods. Previous proposed approaches focus primary on grasping and use constraints to define certain interactions with significant limitations in contrast to our analytic model. We also support multiple hands interacting on the same virtual objects. Our system identifies contact points that lie at both the hand surface and the object surface and is capable of updating these on the object's surface during contact. The user can control the magnitudes of the forces applied to the objects to induce natural interactions like in reality. Our proposed method sufficiently solves the penetration problem which represents a major challenge in hand interaction environments. In addition to that we show a simple method to visualize reasonable hand-meshes during interaction. We also show how to address hand visualizations for sufficient depth perception within *AR* and *VR* environments. We evaluate our approach with a pilot study which demonstrates that our method is perceived more realistic and natural, and allows for more diverse interactions. Further, we evaluate the computational complexity of our method to show real-time performance in *AR* and *VR* environments. An accuracy evaluation for the task of virtual object placement shows that our method enables more accurate interactions than previous methods.

Kurzfassung

Wir stellen eine physikbasierte und performante Methode für Handinteraktionen im Bereich von *AR* und *VR* vor. Unsere Methode basiert auf dem Coulomb-Reibungsmodell welches das komplexe Phänomen der Reibungskraft aus der Realität approximiert. Wir zeigen, wie dieses Modell effizient in einer Grafik Engine für Echtzeit-Performance implementiert werden kann. Dieses Modell ermöglicht uneingeschränkte und sehr überzeugende Simulationen vieler Arten von Aktionen wie Schieben, Ziehen, Greifen oder sogar geschickte Manipulationen wie das Drehen von Objekten zwischen den Fingern. Für die Interaktionsmöglichkeiten gibt es keine Limitierungen im Bezug auf Objekte oder Handpositionen. Da es sich um ein analytisches Modell handelt, benötigen wir im Gegensatz zu früheren Methoden keine vorgefertigten Daten. Bisher vorgestellte Ansätze konzentrieren sich primär auf das Greifen von Objekten und weisen deutliche Einschränkungen von Interaktionsmöglichkeiten auf. Wir unterstützen auch mehrere Hände, die gleichzeitig mit denselben virtuellen Objekten interagieren können. Unser System identifiziert Kontaktpunkte, die sowohl an der Handoberfläche als auch an der Objektoberfläche liegen. Der Benutzer kann die Magnitude der auf die Objekte angewendeten Kräfte kontrollieren, um natürliche Interaktionen auszuführen. Unsere vorgeschlagene Methode löst effizient das Penetrationsproblem, welches als große Herausforderung im Bereich von virtuellen Handinteraktionen gilt. Daneben stellen wir eine effiziente Lösung vor, um Kontaktpunkte auf der Oberfläche während einer Objektmanipulierung neu zu bestimmen. Darüber hinaus zeigen wir eine einfache Methode, um vernünftige Hand-Meshes während der Interaktion zu visualisieren. Wir zeigen auch, wie man Handvisualisierungen für eine ausreichende Tiefenwahrnehmung innerhalb von *AR* und *VR* Umgebungen nutzen kann. Eine Pilotstudie zeigt, dass unsere Methode realistischer und natürlicher wahrgenommen wird und vielfältigere

Interaktionen ermöglicht. Da Performance eine der wichtigsten Kriterien ist für bereits aufwändige Umgebungen wie *AR* und *VR*, zeigen wir anhand einer Performance Analyse dass unsere Methode sehr effizient ist. Wir bewerten unsere Methode ebenso quantitativ für die Aufgabe der virtuellen Objektplatzierung und zeigen, dass unsere Methode genauere Interaktionen ermöglicht als bisherige Methoden.

Contents

List of Figures

List of Tables

1 Introduction

With the rise of Virtual Reality (VR) and Augmented Reality (AR) or Mixed Reality (MR), the gap between our real world and the virtual world is fading and brings up whole new potential ways of interactions. While *VR* connects the human senses to artificial realms, *AR* follows the opposite path to enhance reality with virtual objects. However, both share a similar goal, namely immersion. Having access to these technologies requires to reconsider current computer interaction systems entirely. It drives us to form new ways of more natural and realistic interaction possibilities which can negate the need of learning interaction methods with artificial controllers. We can create new ways of interactions using our bare hands like [2, 17, 46], however the State-of-the-Art (SotA) is still far away from realistic and natural interaction systems considering the existing devices such as the Oculus Rift, HTC Vive, and the HoloLens.

VR and *AR* applications both require interactions between the user and virtual elements. Controllers such as the Nintendo Wii-Mote [47], Oculus Touch, HTC Vive Controller, or HoloLens Clicker can be used. However, they are cumbersome and not very intuitive to use, and the possibility of using our bare hands for the interaction is very appealing for games and all kinds of *VR* and *AR* environments. It can even play a key role in medical applications [8].

Many methods and hardware [15, 32, 54] have been developed to accurately capture the 3D pose of the user's hands. This is, however, still not sufficient for realistic hand interactions: If the user's hand is simply modeled as a 'kinematic object', the virtual hand follows the tracking information for the real hand motion and can penetrate the virtual objects, which can lead to very unstable results in physics engines [17]. The Leap Motion Interaction Engine[1] does solve this penetration problem and supports grasping but in a non-physics-based way as the hand motions are directly transferred to the held objects.

[1] `https://developer.leapmotion.com/unity/`

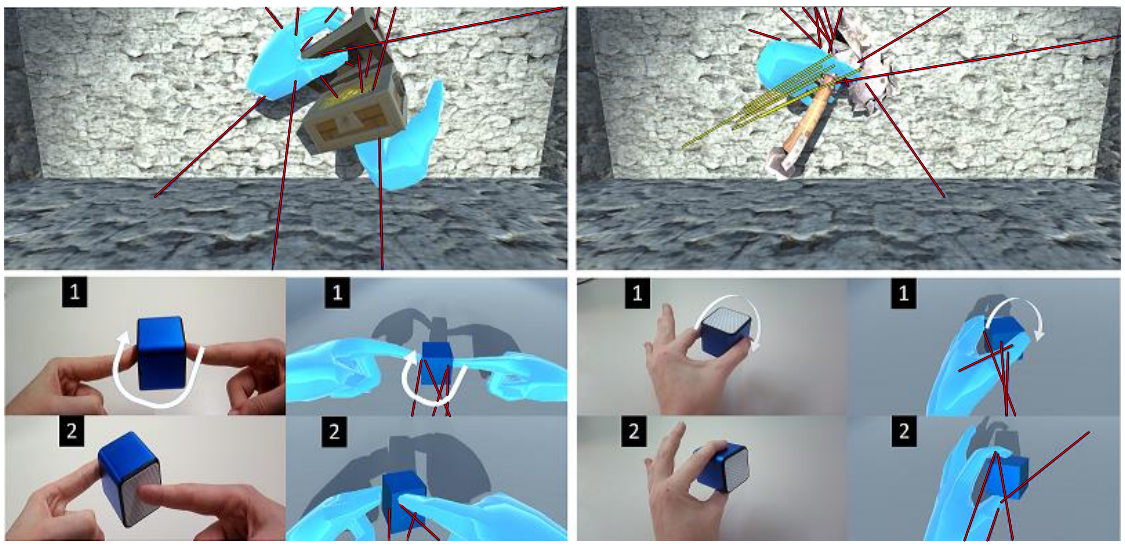


Figure 1.1: Demonstration of our physics-based hand interaction method within *VR* environments. Our approach allows countless complex interactions such as handling a box in one hand and opening it with the other (top-left), or smashing a wall with an axe held in one hand (top-right). We can even spin an object between two fingers from two different hands (bottom-left) or from the same hand (bottom-right) in a very realistic manner. The red arrows indicate the contact forces calculated with our method and applied to the objects. Yellow arrows illustrate the global hand motion on surface points.

To be convincing, the physical interaction between the hands and the virtual objects needs to be realistically simulated. Because such simulation can be very complex, most proposed interaction approaches are simplified, constrained, or artificial. For example, some methods recognize gestures that trigger some predefined interactions [6, 28, 48]. Other methods require some prior training phase [38, 40]. Physics-based methods, like ours, provide more unconstrained interaction possibilities [3, 17, 20, 31]. However, currently, these methods are mostly limited to simple grasping interactions.

In reality, our ability to interact with objects is due to the presence of friction between the surfaces of the objects and our hands, as friction is a force that resists motion. Generally spoken, the phenomena of friction is the sole explanation why interactions between objects are possible at all. Therefore we have aimed to take friction forces into account for our approach.

In this thesis, we propose to use the *Coulomb Friction Model* for properly simulating the forces which the user applies to the objects according to the 3D pose. The Coulomb friction model captures under the same framework both static and dynamic effects that can occur when two objects are in contact. It is only an approximation of real dynamics, but as we will show, it produces convincing motions while remaining tractable, in contrast

to approaches such as [45], which rely on more complex physics models but are much less efficient. In other words, we argue that the *Coulomb Friction Model* is a good trade-off between realism and tractability for interaction in *VR* and *AR*.

This model choice indeed allows us to simulate interactions much more complex than ordinary grasping. For example, with our approach, the user can push, pull, let an object slide along his/her hand if (s)he does not grasp it firmly enough, or even spin an object between two fingers. Fig. 1.1 illustrates various types of interactions possible with our approach.

Our method starts from the 3D hand pose—in practice we use the Leap Motion device to estimate it. Using a simplified 3D model of the hand, we detect the intersections between this hand and the virtual objects. From these intersections, we define contact points between the virtual object and a virtual hand model, which corresponds to the real hand just before it starts penetrating the virtual objects. As the hand can still slide along the objects' surface during contact, we present a solution how to efficiently update new contact points on the surface while the hand is already interacting with a virtual object. Further we explain how we approximate a property of the human skin to generate multiple contact points for each contact. Then we show how to apply the *Coulomb Friction Model* to the contact points, taking into account the force applied by the user which we take proportional to how much his/her real hand penetrates the virtual objects and the friction parameters of the materials of the objects. The forces which are exerted from the hand model to the object are called contact forces. The computed contact forces counteract external forces applied to the objects such as gravity or collision forces. As a next step we show how to introduce the contact forces in a physics engine such as PhysX, which simulates the objects' motions.

1.1 Outline

In Chapter 2 of the paper, we first give a short overview over related work. We classify proposed hand interaction methods in 4 different classes, starting from kinematic gesture-based, over heuristic-based methods, to physics-based approaches and finally hybrid methods. We explain aspects and properties of each different class, discuss several approaches in detail and point out the drawbacks of previous approaches from recent years.

In Chapter 3 we present our method in detail. Section 3.1 illustrates our area of contribution. Section 3.2 explains some notations and conventions. Important arrangements like our hardware setup for *VR* as well as for *AR* are covered in Section 3.3. In Section 3.4 we talk about the phenomena of friction which represents the basement for the overall idea of our method. Section 3.5 covers the estimation of the contact points between the

virtual objects and the virtual hand, how we update the surface points during contact, and how we calculate the forces that are applied to the virtual objects for realistic physics simulation. In Section 3.6 we discuss some visualization aspects for the virtual hand mesh in *VR* and *AR* environments. Section 3.7 covers important implementation details of critical aspects. In Section 3.8 we explain some notable coordinate transformations from the Leap Motion and alignment for correct hand perception with stereoscopic rendering using image pass-through from the controller. Finally, our first simple kinematic grasping approach is discussed in Section 3.9.

Chapter 4 presents qualitative results of our method applied to several interaction tasks. We present different object manipulations like pushing, pulling, grasping, lifting, palmar grasping, sliding, tilting, spinning, and multi-hand interactions. The results are shown for *VR*- as well as for *AR* environments.

In Chapter 5 we present the results of a pilot study and a performance analysis with different configurations demonstrating that our proposed method is capable of stable real-time performance. Additionally, we present a quantitative comparison with other interaction approaches for a positioning task that requires subtle interaction capabilities.

Chapter 6 contains our conclusions and potential improvements. We reflect our results and discuss how our approach opens new possibilities for future work which could build upon our approach.

2 Related Work

3D interactions with virtual objects have recently become an active research field in Human Computer Interaction and Computer Graphics, with the introduction of 3D interaction systems [16, 34, 42, 46] and vast improvements of Augmented Reality (AR) and Virtual Reality (VR) technologies over the last couple of years [1–3, 17, 20, 23, 24, 31, 37, 48, 53].

Recent kinematic skeleton tracking sensors such as Microsoft Kinect or Leap Motion made research on real-time human interaction systems much more accessible. The Kinect sensor enables easier hand-gesture and human-activity recognition [52]. Empirical work [15] has shown that the Leap Motion is a reliable and accurate system for hand skeleton tracking, which recently got improved with a new Orion tracking software[1], specifically developed for *VR* environments.

While these two sensors provide good results for hand tracking, research has since focused on hand interaction methods. We structure these methods in four different categories to give a more detailed overview of hand-based 3D interaction systems within virtual environments.

2.1 Kinematic Gesture-Based Approaches

A frequent and simple type of hand interaction methods relies on kinematic- and gesture-based interfaces, where a predefined set of gestures is used to perform certain assigned actions. Common gestures include circle, swipe, pinch, screen tap, and key tap gestures [5, 6, 48]. A pinch detection is often used for detecting intended grasp interactions [28]. One popular example of such a gesture-based interface is the Microsoft HoloLens [1, 48]. A few predefined hand gestures are detected in real-time to trigger certain interactions with

[1] `https://developer.leapmotion.com/orion/`

virtual objects, similar to mouse clicks.

Leap Motion provides a device based on infrared stereo cameras that captures 3D hand poses. The company recently released Interaction Engine [2], which also makes use of kinematic grasps to perform simple object transformations. Due to the constrained action space, this approach is also appealing for Machine Learning, where [29, 39] for example detect hand gestures for applications such as free hand control of automotive interfaces. Figure 2.1 shows dynamic- and static hand gestures.

While these interaction approaches make sense for certain *AR* and *VR* environments, the interaction is artificial, highly limited in their possibilities, and thus not sufficient for direct object manipulation.

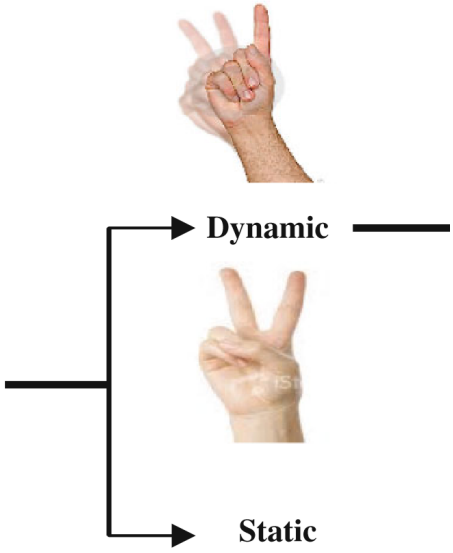


Figure 2.1: Kinematic gesture detection. Detecting dynamic and static gestures to activate interactions. © [39]

2.2 Heuristic-Based Approaches

Heuristic-based approaches require predefined heuristics or *a priori* information about the hand and/or object to perform a limited set of object interactions, mostly object grasping [25, 27, 38, 40]. A data-driven grasp method needs to synthesize prerecorded, real hand data to identify the most similar one during run time from a predefined database [25, 27, 40].

[2] `https://developer.leapmotion.com/unity/`

Object interaction is therefore only possible with object shapes [27, 40] or object categories [25] that are predefined. These requirements induce significant drawbacks, including the need for prior information of either hand and finger poses and/or object shapes, and limitations given by interaction restrictions when synthesizing real-time hand poses from the prerecorded grasp database. This significantly limits the practical application of such methods in unconstrained environments. Figure 2.2 illustrates the synthetic process to find the closest match of pre-recorded data with respect to real hand tracking hand pose.

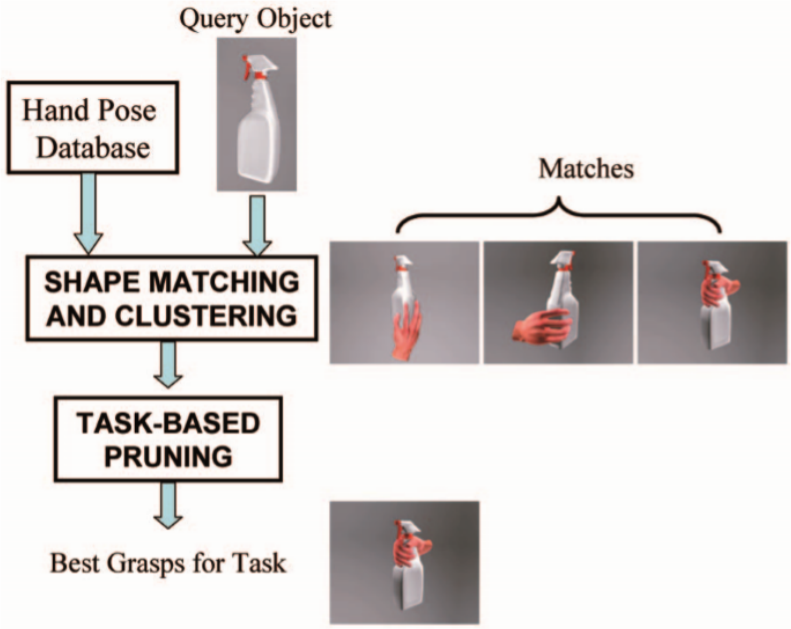


Figure 2.2: Synthesizing real hand tracking with pre-recorded object shapes and corresponding interaction pose. © [25]

2.3 Physics-Based Approaches

Fully physics-based hand interaction systems were recently not widely researched due to the complexity and challenges, such as speed and stability of the physics simulation [9], or accuracy of hand tracking [44]. Hilliges et al. [17] showed with HoloDesk an entirely physics-based interaction system using particles on the hand mesh that induce forces for object grasping. The method suffers from occlusion problems due to the tracking hardware especially when grasps are performed. A finger-based deformable soft body approach on convex objects using soft pads on rigid finger bones has been proposed by [20]. Limitations are due to the missing palm enabling only finger-based interactions and powerful grasps can cause the soft pads to collapse.

An interesting approach has been proposed by [43] which relies on the more complex *Coulomb-Contensou Friction Model*. Despite the accurate results, the computational complexity of this model is very high, which limits its applicability in *AR* and *VR* where real-time performance is required. A more realistic physical simulation of the hand deformations was proposed by [12, 35] who propose to simulate the hand material (flesh), however, these approaches are computationally expensive and themselves require approximations.

Another unconstrained grasping method has been proposed by [3] using a spring-damper model and haptic feedback gloves. This approach has limitations due to the reduced degrees-of-freedom per fingertip caused by the gloves. Also, the exerted forces are only applied with respect to the fingertips, which limits the grasping possibilities. Another physics-based grasping algorithm introduced by [31] uses a second dynamic proxy hand. However, since the physics forces are applied through the proxy hand that gets frozen after a contact occurred, the amount of interaction possibilities besides grasping and pushing are not clear and rather limited. Figure 2.3 shows the second rigid hand carrying out simple physics-based interactions like pushing.

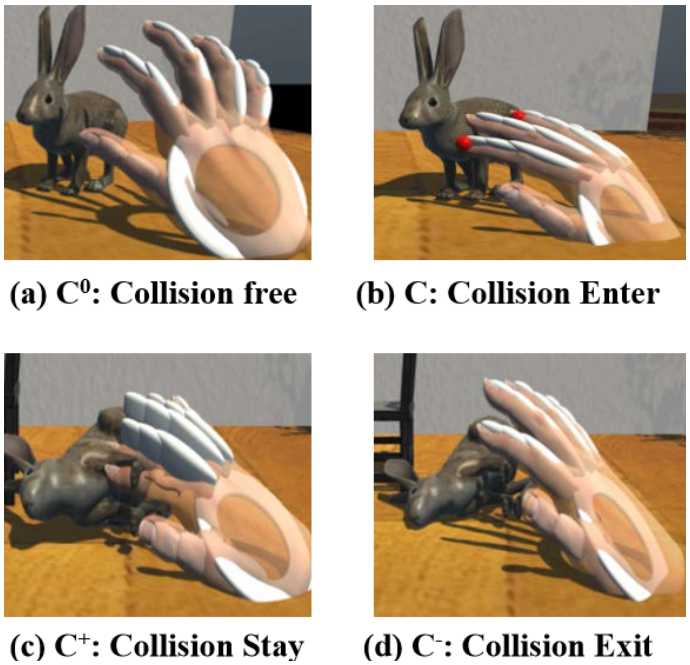
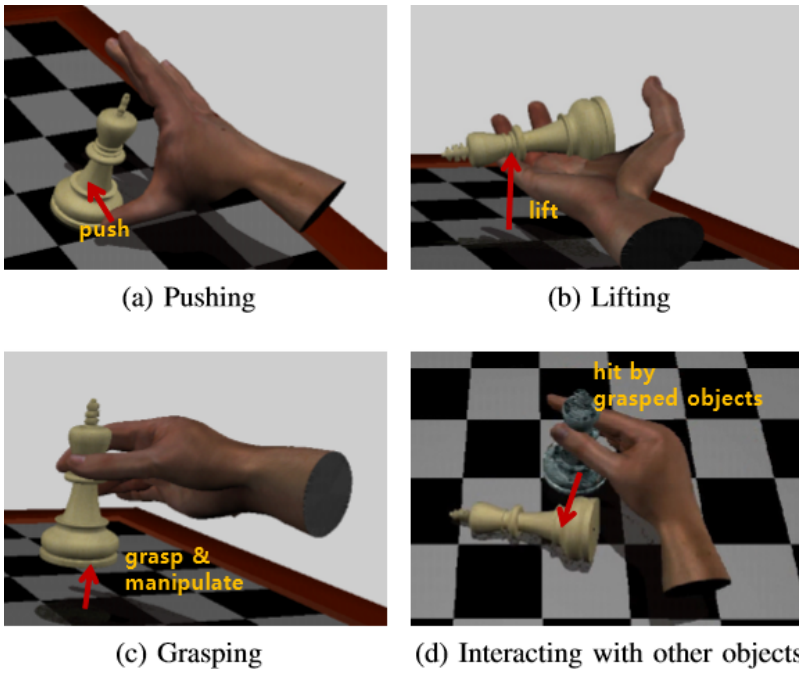


Figure 2.3: Recent physics-based approach using a second proxy hand. A simple push interaction is illustrated. © [31]

2.4 Hybrid Approaches

We consider several methods as a hybrid category since they use certain aspects from different hand interaction categories and combine them together. A recent approach computes initial contact forces via small particles sampled across the virtual skin mesh [23, 24].

However, as soon as an *a priori* defined force threshold is reached, the virtual object is considered as grasped and simply set to kinematic, thus following the virtual hand with global translation and rotation. Figure 2.4 shows results from [23].



(a) Pushing (b) Lifting

(c) Grasping (d) Interacting with other objects

Figure 2.4: Recent hybrid approach mixing computed forces with kinematic transformations. © [23]

Other approaches use a mixture of previously recorded grasps and synthesize new grasps with a computer graphics physics-driven virtual hand [37, 53]. Since these methods rely on prior knowledge or a database of grasps, interactions are still limited.

2.5 Discussion

Comparing our physics-based hand interaction system to these related works, our method is entirely physics-based and fully unconstrained in its nature. Besides having such an unconstrained property, our algorithm also supports stable motion controls similar to methods that focus primarily on grasping [53].

Therefore, we explicitly model exerted contact forces on contact points of the virtual hand bones. We do not rely on any predefined data, states, or conditions, which allows us to perform unconstrained hand and finger interactions known from the real world with arbitrary virtual objects. Our method also allows for finger dexterity and thus unconstrained and dexterous 3D object manipulation during interaction.

We show that the *Coulomb Friction Model* produces realistic hand-object manipulations with proper implementation, which makes it very appealing for computational complex environments with stereoscopic rendering where stable real time Frames per Second (FPS) is arguably one of the most critical aspects.

3 Method

In this section we give a detailed description of our algorithm and explain the underlying physical principles that we use for our approach. Before starting with our physics-based method, we talk about our contribution, introduce the mathematical notations which are used throughout the thesis, and explain notable arrangements with respect to the hardware.

To get familiar with the penetration problem we took a closer look on another very recent physics-based hand-interaction method [31]. It was interesting to see how the penetration problem could be solved in different ways. Recent approaches helped us to get familiar with the overall topic and understand the challenges and possible improvements. We could analyze the drawbacks and limitations of current hand-interaction approaches which led us in the end to our novel physics-based model.

3.1 Contribution

We have designed and implemented our approach strongly with respect to immersive Augmented Reality (AR) and Virtual Reality (VR) environments. We have integrated our system in both *AR* and *VR* setups with according State-of-the-Art (SotA) hardware to ensure a high degree of immersion and presence. However, our system is still a general stand-alone hand-interaction approach and thus also usable for desktop setups and other environments. Figure 3.1 illustrates the area of our contribution.

As discussed in the Introduction section, our contribution lies in physics-based hand-interaction approaches for direct virtual object manipulation. We introduce a novel and efficient implementation of the *Coulomb Friction Model* for hand interaction approaches and thus focus primary on the methodology and implementation itself. For the sake of our main contribution we have not added any haptic feedback system to improve usability.

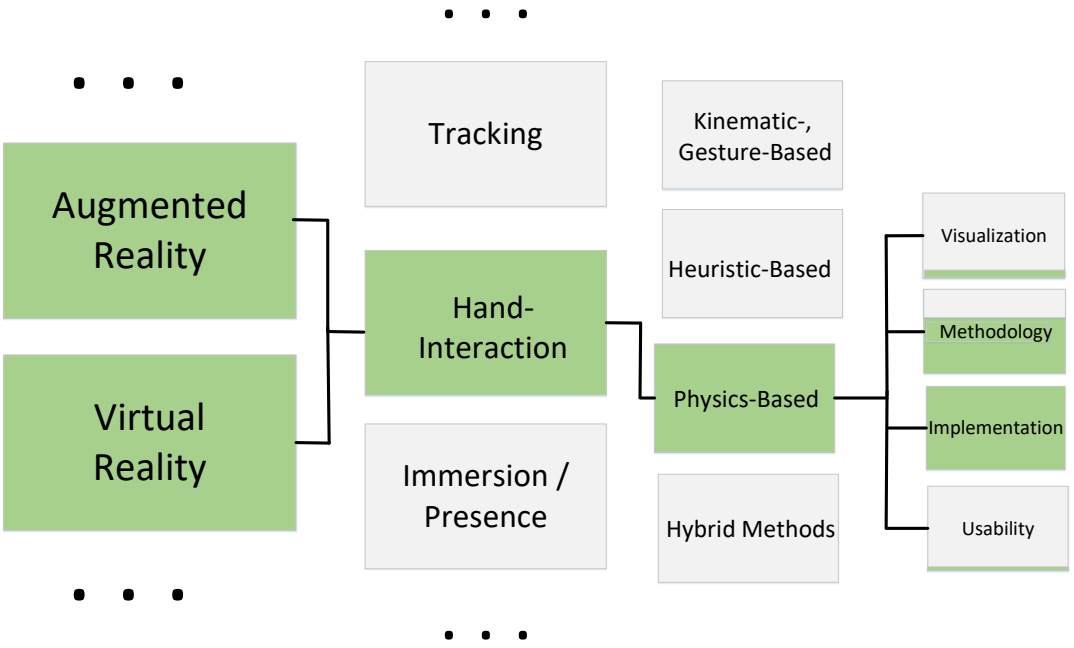


Figure 3.1: Area of contribution. Rectangles are filled with respect to the contribution.

We expect that integrating a haptic feedback system would improve usability significantly, however it would not change anything in regard to the methodology or functionality.

3.2 Notation and Conventions

Scalar values are represented by italic fonts, e.g., x or c_i. Matrices and vectors are depicted in bold font, e.g., $\mathbf{M}$ or $\mathbf{V}$. Vector spaces are depicted in double-lined upper case letters, e.g., $\mathbb{R}^3$ or $\mathbb{Z}^3$. Functions, mapping between different spaces are depicted by an arrow. An overview over the notation is given in Table 3.1.

Entity	Notation
Scalar	a, c_i
Vector in 3D	$\mathbf{V} = (x, y, z)^{\mathrm{T}}$
Matrix	$\mathbf{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
Vector Space	$\mathbb{R}^3$
Mapping Function	$\mathbb{R}^3 \rightarrow \mathbb{R}^2$

Table 3.1: List of notations used in this thesis.

3.3 Arrangements

First, we show our *VR* and *AR* hardware setup. We explain how we configured the Leap Motion for the *VR* and *AR* device to align the virtual hand skeleton as close as possible with the real human hands. Then we show how to identify interactions between the physical user's hand and the virtual objects, and contact points between them. Using these contact points, we calculate the forces that are induced by the hand interaction and apply them to the virtual object. We further show how to implement them efficiently into an existing physics engine such as PhysX. In the subsequent section we cover the visual representation each for *VR* and *AR*.

3.3.1 Hardware Setup

In this section our hardware setup is shown, each for *VR* and *AR* environments. As for *VR*, we have used an Oculus Rift DK2 as shown in Fig. 3.2.

Figure 3.2: Hardware Setup: Leap Motion mounted on Oculus Rift DK2

With a hardware like this we can use proper 3D stereo rendering to achieve immersive *VR* experiences. The process of stereo projection transformation for immersive 3D perception is covered in detail here [18]. To track our real hands we use the Leap Motion sensor which is a Structured Light (SL) system similar to Microsoft's Kinect. *SL* sensors are used for shape reconstruction and pose estimations. Light beams project certain patterns onto the tracked object. Depending on the shape of the object, the light pattern features distortions. With a stereo camera setup it is then possible to measure the projected light pattern and compute the degree of distortions to get an accurate depth perception of the tracked object. Figure 3.3 illustrates the *SL* principle of the Leap Motion sensor.

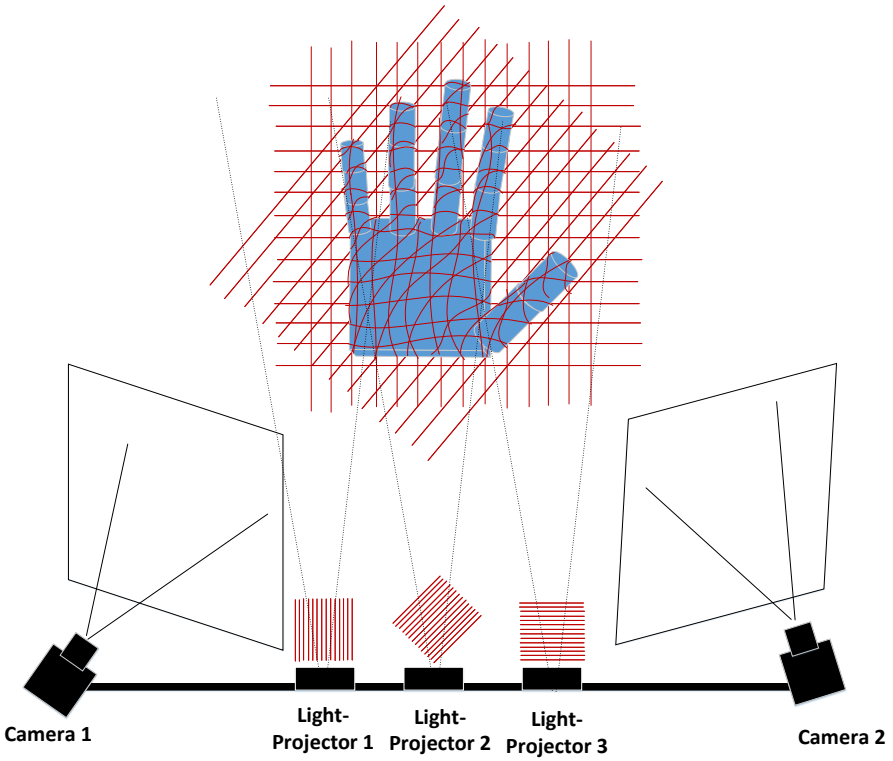


Figure 3.3: Simplified illustration of Leap Motion's *SL* setup to track human hand skeletons.

The Leap Motion is specialized in hand skeleton tracking and recently released a major software update known as Orion. Orion was specially improved for tracking hands in *VR* setups where the Leap Motion sensor projects the structured light patterns in forward direction onto the human hands. The tracking capabilities of Orion has been proven as sufficient. This is especially true considering that the Leap Motion Unity assets allowed us to start with the actual implementation of our method straight on. That is why the Leap Motion with Orion software was a good choice overall.

As for the *AR* setup, we have used Microsoft's HoloLens. This was a rather easy choice since the HoloLens is undoubtedly today's *SotA AR* device. The HoloLens is considered as unrivaled currently, providing impressive real-time tracking and mapping capabilities. However, the limited Field of View (FoV) is a drawback but robust tracking has been a much more essential factor for us. Even the limited *FoV* is sufficient to interact on virtual objects using the whole hand, and even two hands at the same time as far as the virtual object is not too big. Using the Oculus Rift DK2, it was obvious to find a good spot to

mount the Leap Motion on. With the HoloLens the potential mount locations are rather limited since it is an Optical See Through Head Mounted Display (OSTMHD) where the Leap Motion sensor must not occlude the wave guides of the HoloLens. Thus we have mounted it ontop of the HoloLens as shown in Fig. 3.4.

Figure 3.4: Hardware Setup: Leap Motion mounted on HoloLens

That position ensures that we do not occlude any cameras or waveguides of the HoloLens but on the same time guarantee that the FoV of the Leap Motion can fully track both hands when wearing the HoloLens and interacting with virtual objects.

3.3.2 Calibration - Mount Correction

Aligning the virtual hand skeletons with our real hands as close as possible was a critical task. Especially in AR scenes since the user could not only feel where his/her real hands are but also clearly see due to the optical see through waveguides of the HoloLens. However, mismatches in scale due to flawed hand alignment can also be noticed in VR if it is not done correctly. This is because the user still has a very good perception of where his hands are, even if he/she is immersed in a virtual and unable to see the 3D position of his/her hand. These mismatches do not break the immersion, but they do break the presence. To ensure a realistic perception of the virtual hands, we first had to counteract the translation of the physical hardware mount of the Leap Motion for the Oculus Rift DK2 aswell as for the HoloLens and apply this translation to the virtual Leap Motion controller within the graphics engine. The transformation matrix accounting for the mount translation of the physical Leap Motion device is given by:

$$\mathbf{T}_{\text{Mount}} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Oculus Rift DK2 Mount: To match the Leap Motion sensor with the origin of our eyes it was mandatory to apply a translation on the Z-Axis to the virtual stereo cameras. This translation ensures that the user does feel the virtual hands as of his real hands in position. The translation vector for the Oculus Rift DK2 with a Leap Motion mount on the front plate is:

$$\mathbf{V}_{\text{OculusRift}} = (0, 0, 0.08)^{\text{T}}$$

HoloLens Mount: The HoloLens mount shows a different translation for the physical mount as one can see in Figure 3.4. For the HoloLens mount it is important to point out that the correction can vary depending due to the flexible adjustment of the HoloLens device itself. The Leap Motion is ontop of the HoloLens and thus above humans eyes. The translation applied to the virtual Leap Motion controller is:

$$\mathbf{V}_{\text{HoloLens}} = (0, 0.07, 0.05)^{\text{T}}$$

After applying each of the corresponding mount corrections the overall hand alignment was very good for *VR* and *AR* aswell. Figure 3.5 shows the fitting of the virtual hands and the real hands after the applied HoloLens mount correction. The images have been taken using the HoloLens hardware setup shown in Figure 3.4.

Figure 3.5: Alignment with virtual hands and real hands. The skin colored mesh aligns very close with the real hands which are consequentially barely visible. Different hand poses are presented with both hands simultaneously to show the fitting alignment of real- and rendered hands.

3.4 Phenomena of Friction

The phenomena of friction represents the basement of our entire approach. In this section we give a short overview over the complexity of the friction phenomena and summarize different introduced friction models. In the following subsections we refer to the references [7, 10, 11, 22] as the main source of information about the phenomena of friction and all the related friction models.

3.4.1 Principles of Friction

Friction is an extremely complex phenomena. It is the reaction force between two surfaces in contact. Over the past years several models have been introduced to simulate friction as close to real friction as possible. In reality, friction is dependent on many components such as the surface materials, temperature, wear, the topology of the objects, relative velocity, area of contact, or the presence of lubrication.

Without the phenomena of friction, we would not be able to grasp objects or interact with objects in any meaningful way. In reality, all object interactions depend on the phenomena of friction and would not be possible without it.

In the following subsection we discuss several proposed friction models found in the literature.

3.4.2 Friction Models

The *Coulomb Friction Model* is by far the most well-known model to approximate the friction phenomena. It is also known as classic friction model due to its fame. It is a rather simple yet good approximation of the whole complexity of the real friction phenomena, and is widely used in the fields of engineering.

The Coulomb model states that the magnitude of kinetic friction is independent of the velocity. Thus the model does not take into account the speed when objects are in a dynamic state of relative motion. The model is built upon *Amonton's Law of Friction*. The law states that the magnitude of the friction force between two materials is directly proportional to the applied normal force with a constant proportionality. Further, it states that the friction coefficient is constant and indipendent of the sliding velocity or contact area. The Coulomb model is described as:

$$F_{friction} = \mu F_N, \tag{3.1}$$

where $F_{friction}$ is the friction force, μ is the material dependent friction coefficient describing the friction between the objects in contact and F_N is the induced pressure.

The *Viscous Friction* takes into account the speed of the objects in contact. It is often combined with the *Coulomb Friction Model*. Thus, the model expresses a non-linear velocity dependency:

$$F_{friction} = F_v v \tag{3.2}$$

According to Stribeck [41], the velocity dependence is continuous and thus is not decreasing discontinuously. This is called the *Stribeck Effect*.

The *Karnopp Model* counteracts the disadvantage of zero velocity and shows a better adjustment when switching between different contact states. However, one drawback of the model is that it uses external forces as input for the equation which are often not explicitly given [7].

Another static model, which accounts for dynamic friction phenomena is the *Armstrong Model*. The model introduces temporal dependencies for static friction and the *Stribeck Effect* and consists of two separate models, one for the sticking- and one for the sliding contact state.

All of the models so far are static models. The *Dahl Model* is the first dynamic model which we discuss here. Dynamic models have the property of better friction compensation. This particular dynamic model is a generalisation of the ordinary *Coulomb Friction Model*. One of his findings was that bearing friction and solid friction have similar behavior. According to this model, the friction force is only dependent on the position. It does not capture the *Stribeck Effect* nor does it capture static friction.

The *Bristle Model* introduces the behaviour of microscopical contact points between two surfaces in contact. The number of contact points are random due to the irregularities in the surfaces. A bond between flexible bristles model each point of contact. The strain in the bond increases as the surfaces move relative to each other and the bristles act as springs. The modeled force is described by the following equation:

$$F_{friction} = \sum_{i=1}^{N} \sigma_0(\mathbf{x}_i - \mathbf{b}_i), \tag{3.3}$$

where N describes the total number of bristles, σ_0 the stiffness of the bristles, $\mathbf{x}_i$ the bristle position and $\mathbf{b}_i$ original bond formation point. Compared to the other models, this model captures the random nature of friction which depends on the number of bristles. However, due to the high complexity of this model it is considered as inefficient in simulations.

As an computational more efficient approach, the *Reset Integrator Model* keeps the bond constant which limits the increase of the strain. The limitation here is given by the

point of rupture. While it is simpler to simulate than the *Bristle Model*, it shows some discontinuities.

Another dynamic model is the *Bliman-Sorine Model* which emphasizes the importance of rate independence. It is closely related to the *Dahl Model*. This states that the friction force is not explicitly dependent on the velocity but considers the distance traveled after a velocity zero crossing. Thus, the distance can be taken into the equation instead of time as an independent variable.

The *LuGre Model* is the second dynamic model being closely related to the *Dahl Model*. It is inspired by the bristle approach while additionally considering lubrication effects. Bristles will deflect if tangential force is applied. The friction is modeled as the average deflection force of elastic springs.

With these models we have covered various significant models how to possibly capture the phenomena of friction. While the literature introduced many complex models how to approximate the phenomena of friction in more detail, all of the proposed friction models build upon the *Coulomb Friction Model*, i.e., despite all complexity of later introduced models, the important Coulomb friction component is always present in every proposed complex model.

As we will show with our approach, the Coulomb model is despite its simplicity a good approximation of the friction phenomena which can produce realistic physics-based hand-object interactions while the computational complexity is rather low. This is a significant advantage for performance concerns in already expensive computational environments such as *AR* and *VR*. With our method, we show that this model can be used to simulate realistic hand-object interactions for computational expensive environments in *VR* and *AR*.

3.5 Physics-Based Hand-Object Interaction

In this section we give a detailed description of our algorithm and explain how we can make use of the phenomena of friction for our approach. First, we show the virtual skeletal hand model which is synchronized with the physical user's hands. Next, we explain how to identify interactions between the hand skeleton and the virtual objects, and estimate contact points between them. We present an efficient solution how to update surface points during contact. Using these contact points, we calculate the forces that are induced by the hand interaction and apply them to the virtual object. We further show how to implement them efficiently into an existing physics engine.

3.5.1 Skeletal Hand Model

For our approach we rely on the simplified skeletal hand model shown in Fig. 3.6, which is made of simple shapes: Three capsules per finger and one cuboid for the palm. Each of these shapes can have one contact with the object. In practice, we continuously track the 3D pose of this hand model in terms of 3D location and orientation of each bone (phalanx) using the Leap Motion device.

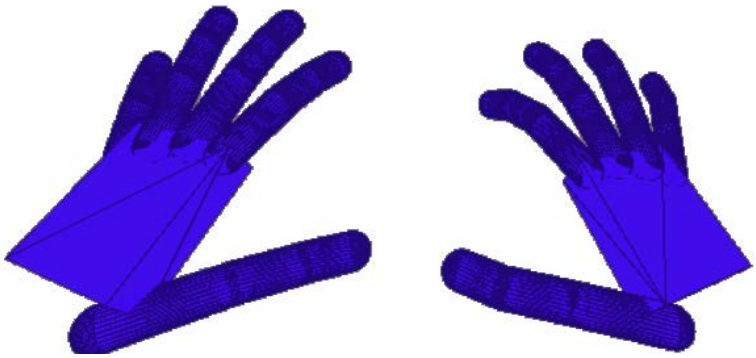


Figure 3.6: The skeletal hand model which we use to detect collision between the user's hand and the virtual objects. The palm is modeled as a cuboid and each finger consists of three capsules.

It is important to mention that we avoid any real physical contact between the skeletal hand model and the virtual objects due to the interpenetration problem which is further explained below. We handle the contacts between the hand model and objects entirely by ourselves by inducing forces computed with our analytic model. The skeletal hand model is solely used to detect potential collisions and interactions on different parts of the skeleton.

3.5.2 Hand-Object Contact Point Estimation

In order to compute the forces which are applied by the hands to the virtual objects, we first need to identify the *contact points* between the user's real hand and the virtual objects. As we mentioned in the introduction, hand-object interpenetration can become a significant problem in physics engines, and we explain here how we efficiently solve the interpenetration problem when estimating contact points between the virtual hand model and the virtual objects.

A contact point is defined as a 3D point that lies on the surface of the virtual object and corresponds to a phalanx of the hand skeleton during interaction.

As the real hand can penetrate the virtual objects, it is not clear how to define meaningful contact points. To do so, we propose the following method: We continuously look for potential collision by defining a small threshold distance around the virtual objects. We create a contact point when a point on the hand model's surface gets closer to a virtual object than this threshold. This event can be detected efficiently using a physics engine. Fig. 3.7 explains how we assign a 3D location to this contact point: Once we detected a point on the hand that is closer to the object surface than the threshold, we estimate a close 3D point on the object surface to this point as the contact point, denoted $\mathbf{C}_i$. We will use this point to compute and apply the forces as explained below.

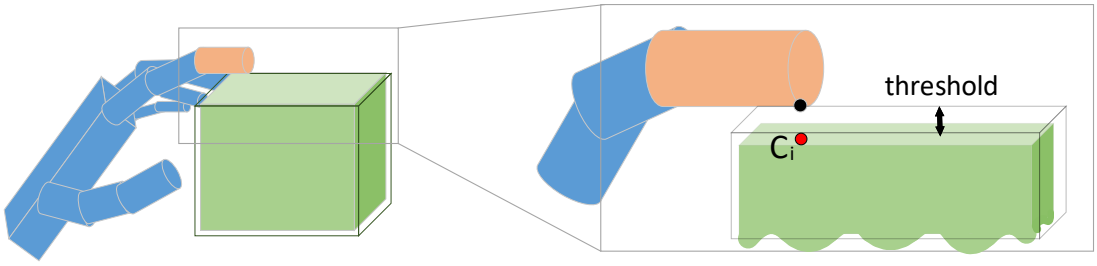


Figure 3.7: Contact point creation. Once we detected a point on the hand that is closer to the object surface than a threshold (the black dot in the Figure), we estimate a close 3D point on the object surface to this point as the contact point (the red dot denoted $\mathbf{C}_i$ in the Figure).

3.5.3 Dynamic Surface Point Updates

Updating the contact points during a contact on the objects' surface represents a significant challenge due to the interpenetrating property of real hand interaction on virtual objects. Updating the contact points is necessary when a contact resides within the dynamic contact state of the friction cone and thus slips along the surface of an object.

A well known method to update contact points on the object's surface is the *God-Object* approach [21, 33]. However, we found a more efficient solution without the need of solving the *Gauss' Projection Problem* or *Unconstrained Acceleration Computation* as in [21].

As we explain below in Section 3.5.5 we distinguish between static- and dynamic states of each contact point. Static state describes stable contacts and dynamic state depicts unstable contacts where the contact point is naturally slipping along the object's surface. As long as a contact remains in the static state there is no need to update the contact point on the surface since the contact is stable.

When we determine a dynamic motion we estimate a new contact point along the ob-

ject's surface. Our solution considers arbitrary object shapes and guarantees that updated contact points remain within a potentially blocked area of the shape. We illustrate the solution in Fig. 3.8. For more details about the voxelization or the surface updates see Section 3.7.

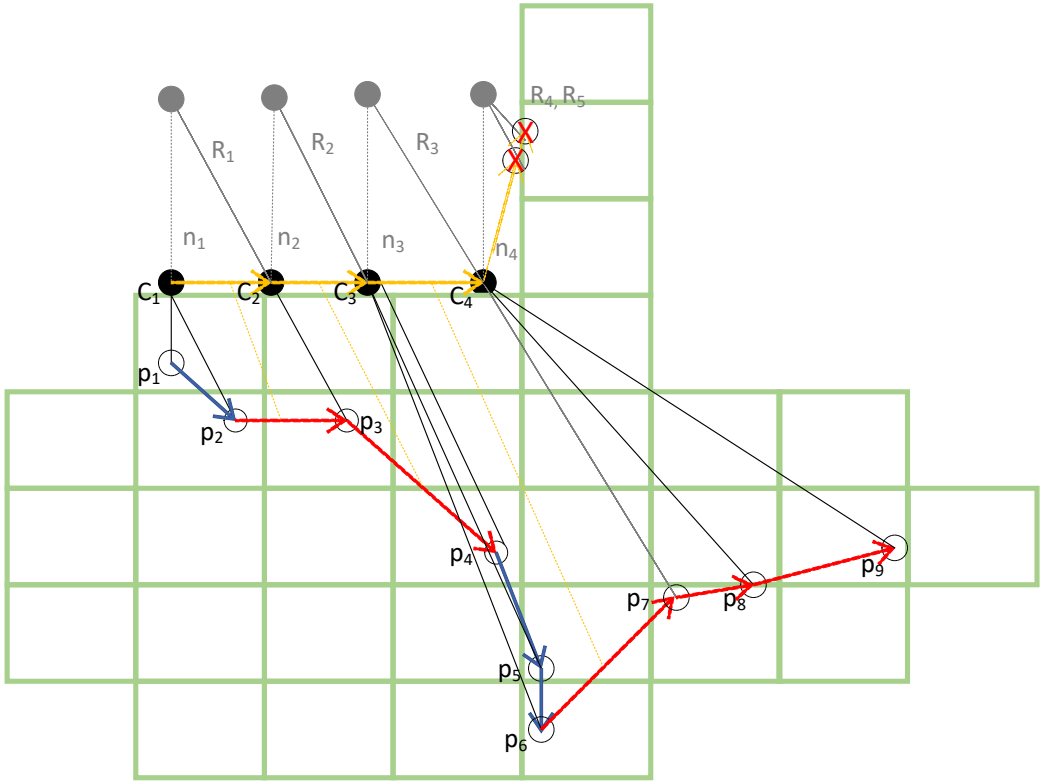


Figure 3.8: Updating dynamic contacts along the object's surface with inverse ray casting. The image illustrates the trajectory of one phalanx inside a 3D voxelized virtual object from top-down view. Blue arrows depict static (stable) motions, red arrows dynamic (slippy) motions and R_n illustrates inverse ray casts in direction of the phalanx $\mathbf{p}_j$.

3.5.4 Skin Approximation

In practice, the human skin is a soft body and forms a contact area rather than a single contact point [14]. This helps to stabilize exerted forces when being in direct contact and keeps control over the object manipulation. We approximate this property by adding contact points sampled on the surfaces of the hand and of the object around the initial contact point. Contacts on the thumb and the palm are sampled over a wider area than the other four fingers. This is due to reason that the thumb and palm have potentially a wider contact area compared to the other 4 fingers.

The number of samples for the patches can be arbitrary high as long as it does not decrease the performance. Stepping from only one contact point to 9 samples for each patch shows already a significant improvement for stabilizing the forces while not sacrificing performance. Figure 3.9 illustrates a multi-sampled patch around an estimated contact point.

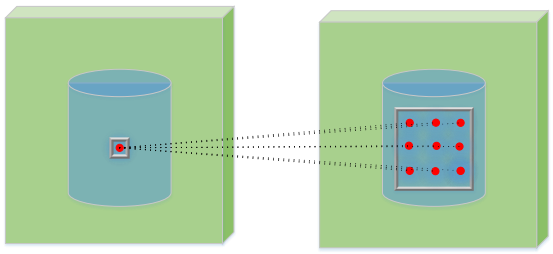


Figure 3.9: Distribution of the applied forces within a contact area. The blue cylinder represents a finger bone and the red dots the actual points where we exert forces. By applying the force on multiple instead of one position, the interaction is more stable and more realistic.

3.5.5 Contact Force Analysis

Our method considers the normal and tangential component of the contact force which is directly exerted according to the motion derived from the real hand tracking data. The contact force is computed for every bone in contact with an object and applies to the corresponding contact patch.

As we will explain, the *Coulomb Friction Model* computes the tangential friction forces from the tangential component of these forces. The tangential component takes into account the frictions along the surfaces. The model distinguishes between two types of friction: dynamic and static friction. Static friction happens on stable contact points and has usually a higher friction coefficient, whereas dynamic friction occurs when objects are in relative motion, that is, when an object slides along the surface of the hand. The criterion to distinguish between static and dynamic friction forces is based on a 'friction cone', as explained below.

Contact Force. We define the contact force $\mathbf{f}_i^{\text{contact}}$ as:

$$\mathbf{f}_i^{\text{contact}} = \gamma(\mathbf{C}_i - \mathbf{p}_j) \,, \tag{3.4}$$

where $\mathbf{C}_i$ denotes the contact point on the object's surface, and $\mathbf{p}_j$ is the 3D centroid of the phalanx, which is tracked by the Leap Motion sensor. This formulation resembles a spring model, as used in [4] for example. When the user interacts with an object, $\mathbf{p}_j$ intersects the object's volume since the hand model follows the pose estimated for the user's real hand. This formulation is illustrated in Fig. 3.10.

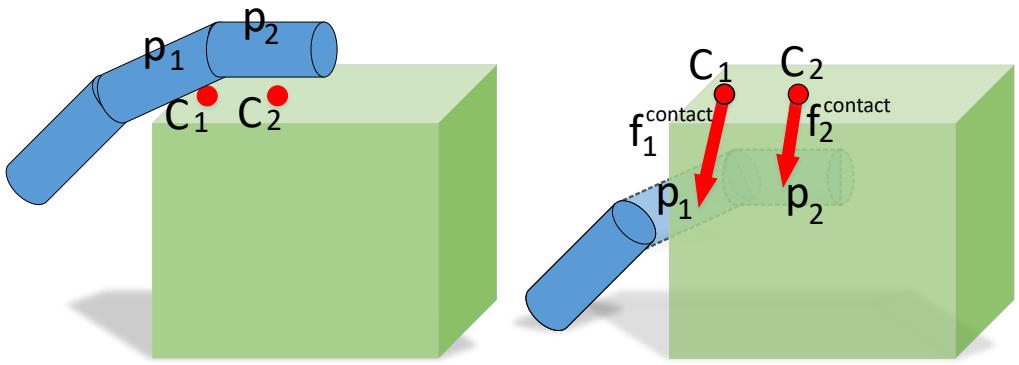


Figure 3.10: Estimating the contact force. The contact force $\mathbf{f}_i^{\text{contact}}$ is proportional to the distance from the contact point $\mathbf{C}_i$ to the current location of the intersecting bone $\mathbf{p}_j$.

For the experiments in this paper, we set $\gamma = 400$. Optionally, we propose a way to auto adjust γ to the physical properties of the object (volume, mass, surface material) at the end of this chapter.

With the expression of Eq. (3.4), the direction of the contact force is dynamically updated according to the bones current position, thus allowing the user to control the force finely.

Normal Force. The normal component of the contact force is calculated as:

$$\mathbf{f}_i^{n\text{-contact}} = (\mathbf{f}_i^{\text{contact}} \cdot \mathbf{n}_i)\mathbf{n}_i \,, \tag{3.5}$$

where $\mathbf{n}_i$ is the surface normal vector of the object mesh at the contact point $\mathbf{C}_i$. The normal component of the contact force is directly applied to the object. The *Coulomb Friction Model* computes the force $\mathbf{f}_i^{T\text{-contact}}$ applied in the tangential direction from the tangential component of the contact force to take into account the frictions along the surface of the object.

The expression of the tangential component of the contact force is:

$$\mathbf{f}_i^{t\text{-contact}} = \mathbf{f}_i^{\text{contact}} - \mathbf{f}_i^{n\text{-contact}} \,. \tag{3.6}$$

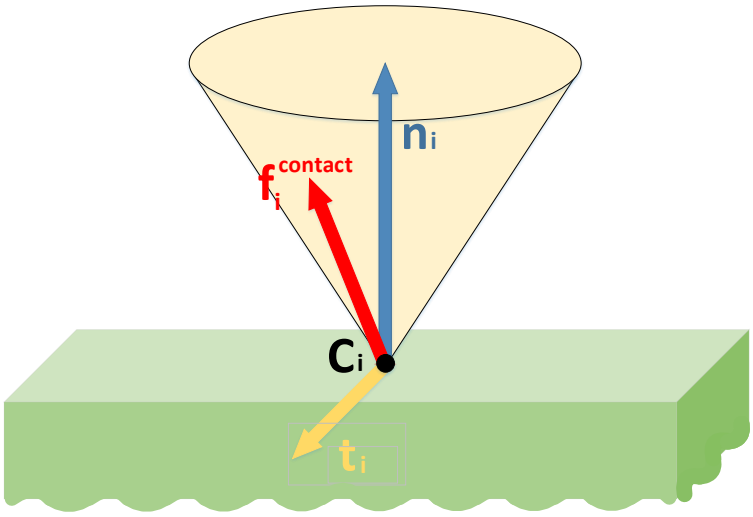


Figure 3.11: Coulomb friction cone for a contact point $\mathbf{C}_i$. The vertex of the cone is located at the contact point and oriented along the surface normal $\mathbf{n}_i$. The contact force $\mathbf{f}_i^{\text{contact}}$ defines the direction $\mathbf{t}_i$ tangential to the surface.

The expression of the tangential force $\mathbf{f}_i^{T\text{-contact}}$ depends whether or not the contact force $\mathbf{f}_i^{\text{contact}}$ is inside the *friction cone*. As shown in Fig. 3.11, the friction cone is defined by the *Coulomb Friction Model* as a cone with the vertex corresponding to the contact point and the axis along the surface normal $\mathbf{n}_i$. The contact force $\mathbf{f}_i^{\text{contact}}$ is inside the friction cone if and only if the following condition is true:

$$F_i^{\text{inside}} = \mathbf{f}_i^{\text{contact}} \cdot \mathbf{n}_i > 0 \quad \wedge \quad \|\mathbf{f}_i^{t\text{-contact}}\|_2 \leq \mu_i^{st}(\mathbf{f}_i^{\text{contact}} \cdot \mathbf{n}_i)\,, \tag{3.7}$$

where μ_i^{st} is the static friction coefficient at the surface location of the contact point $\mathbf{C}_i$.

The tangential force $\mathbf{f}_i^{T\text{-contact}}$ can then finally be computed as:

$$\mathbf{f}_i^{T\text{-contact}} = \begin{cases} \mathbf{f}_i^{t\text{-contact}} & \text{if} \quad F_i^{\text{inside}} \quad \text{is true} \\ \mu_i^{dyn} \cdot \mathbf{f}_i^{t\text{-contact}} & \text{otherwise.} \end{cases} \tag{3.8}$$

This force lets the object slide along the hand surface or on the contrary lets the hand grasp firmly by counteracting gravity, depending on its magnitude and direction, while taking into account the friction properties of this surface. We finally apply the forces $\mathbf{f}_i^{T\text{-contact}}$ and $\mathbf{f}_i^{n\text{-contact}}$ at the patch of contact point $\mathbf{C}_i$.

Figure 3.12 shows the geometrical formulation of the contact force analysis explained above. It is also a 2D projection of the 3D friction cone in figure 3.11.

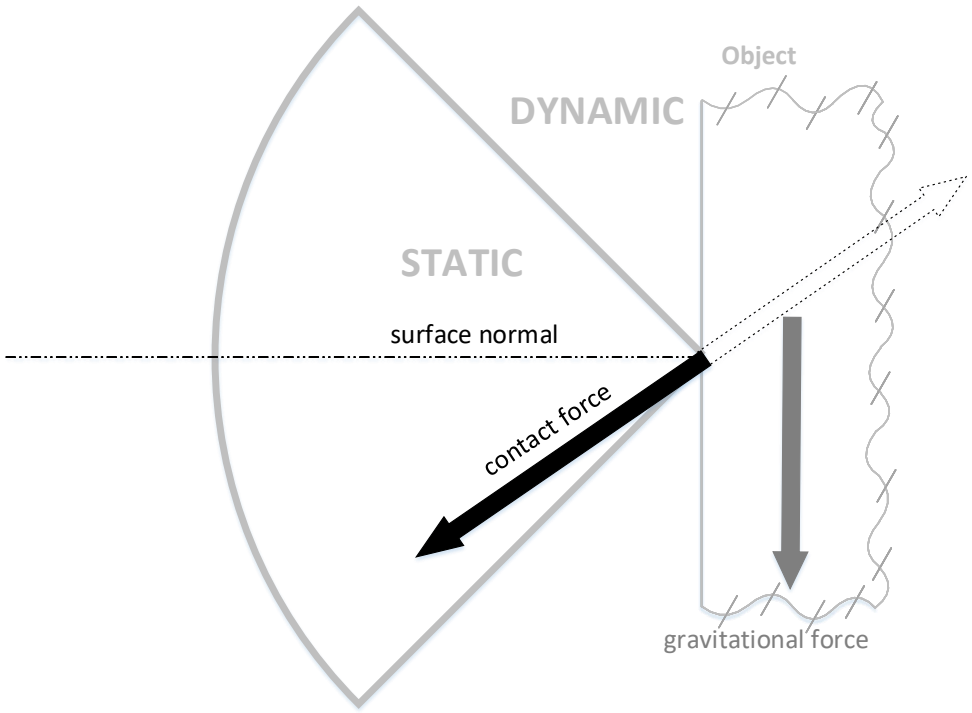


Figure 3.12: 2D projection of the contact force analysis using Coulomb's friction model.

For the material-dependent friction coefficients μ^{st} and μ^{dyn} between skin and other various materials, we use values from experimental data [51].

Tangential Friction Force. The force from the tangential portion of the contact force transfers the tangential motion of the hand onto the object in the direction of the hand movement, whenever the hand is in contact with the object. This is the force that allows the hand to actually move the object due to the increase of the tangential portion in relation to the contact point. External forces such as gravity or collision forces can also indirectly influence the tangential component.

Fig. 3.13 illustrates the importance of the tangential component and allows to firmly pull an object with the hand motion. The hand moves for a small distance within the time interval Δt. This movement induces an increase of the tangential component of the contact force since the bones are moving away in tangential direction from the contact points. If the increase of the tangential component within the time interval Δt is too large in relation to the normal component, the contact force switches to a dynamic state and the contact point starts to slip on the object's surface.

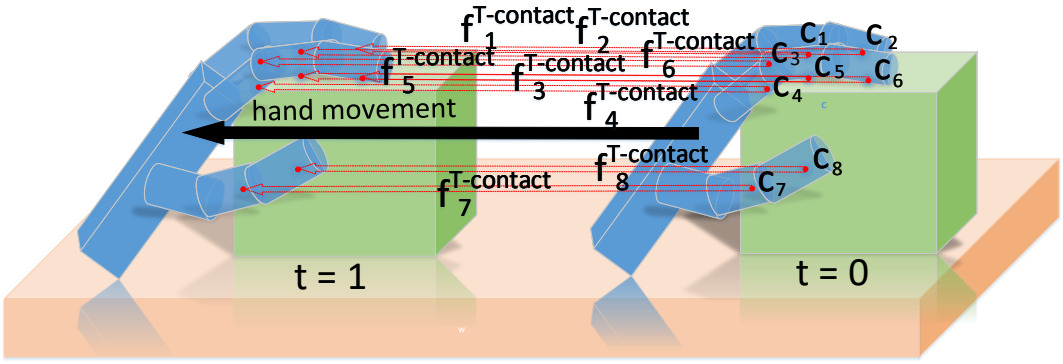


Figure 3.13: Illustration of an increased tangential force $f_i^{T-contact}$ from $t = 0$ over $t = 1$ causing an object to move in the the direction of the global hand motion during contact.

The same principle can occur under the influence of external forces such as gravity or collision forces while maintaining lower contact forces. Fig. 3.14 illustrates gravitational acceleration during dynamic contact states which induce an increase of the tangential portion and thus cause the object to slip between the bones. This happens when the contact was not stable and thus the force resides in the dynamic area as described by the Coulomb model.

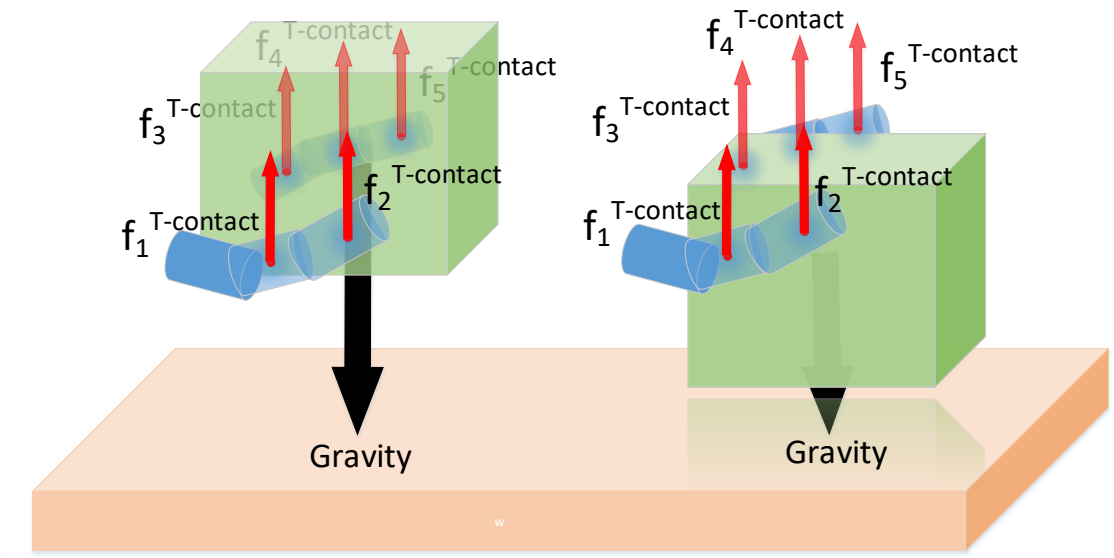


Figure 3.14: Gravitational acceleration as an example for an external force. The black arrow depicts the gravitational force, the red arrows depict the calculated friction forces applied to counteract gravity.

Depending on the contact direction of each contact force, the global hand motion can raise an increase in tangential- or normal direction of each friction cone and thus consequently contribute to its normal portion. This means we consider not only the free finger motion for the contact force but also the global hand motion for the computation of each contact force.

Normal Component Contribution. We show a 2D projection of a friction cone in Fig. 3.15. The Figure shows the friction cone for the case when the hand motion is in the same direction as the initial contact direction of a phalanx. This consequentially increases the normal part of the resulting force, allowing for larger tangential forces according to the *Coulomb Friction Model*, and the friction cone enlarges from the dashed line to the solid line.

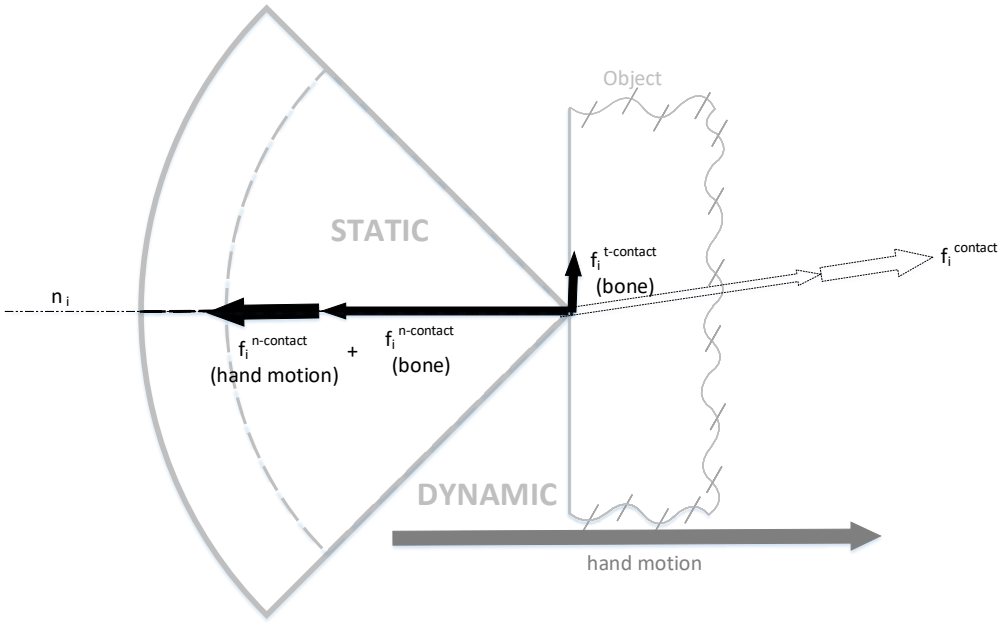


Figure 3.15: Contribution of the global hand motion or external collision force to the normal portion of the established contact. The volume of the friction cone changes with the increased amount of normal force. The contact force is split in normal- and tangential components.

The same applies for collision forces as external forces which would occur in opposite direction of exerted contact forces. The collision force caused by a collision with another object in opposing contact direction would push the grasped object towards the contact point and consequentially increase the magnitude of Eq. 3.4 due to the objects movement. This movement caused by a collision induces an external force $\mathbf{f}_i^{external}$ proportional to the acceleration $\mathbf{a}$ of the object and the object mass m according to the classic relation:

$$\mathbf{f}_i^{external} = m \cdot \mathbf{a} \,. \tag{3.9}$$

Tangential Component Contribution. In Fig. 3.16 we illustrate the case when the global hand motion is in strong tangential direction to the surface. The new resulting contact force shows an increase of the tangential component and does not contribute to the normal component. Thus, it does not change the friction cone, but it increases the tangential component that is applied to the object and decreases the maximum of the allowed tangential portion according to the second expression of Eq. 3.7. Again, the same state can occur as a consequence of external forces. If the gravitational acceleration of

an object is not counterbalanced by sufficient contact force, an object would slip between fingers during a grasp interaction, increasing the tangential component of the dynamic contact state.

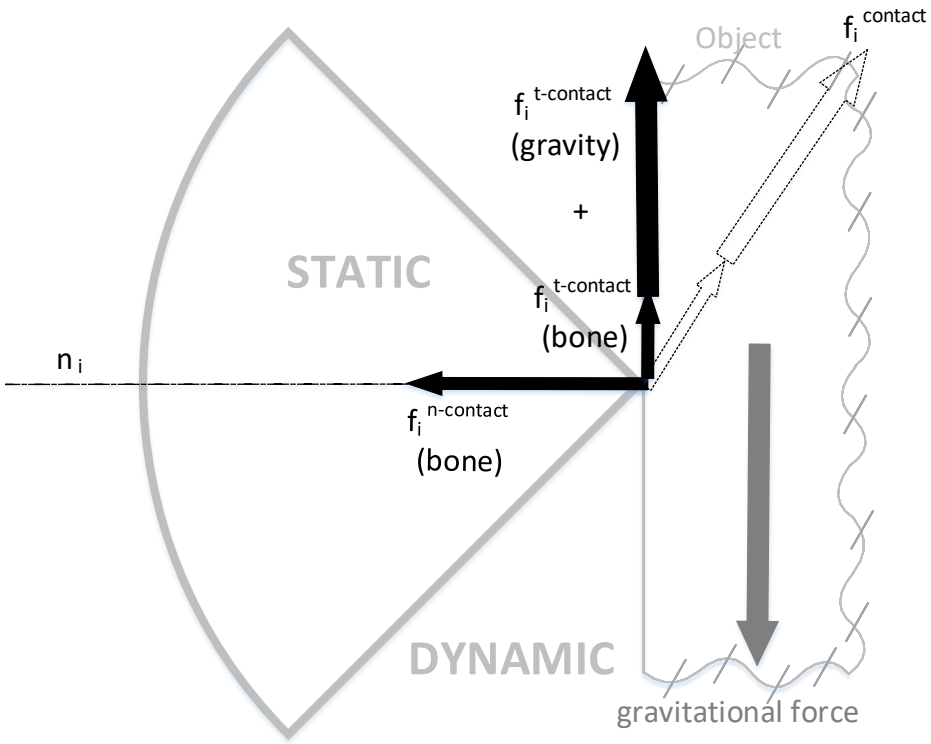


Figure 3.16: Increasing tangential force due to the slipping state caused by gravitational acceleration or global hand motion in tangential direction. The contact force is split in normal- and tangential components.

This analytical part is calculated for every bone of our physical hand model. The total amount of force applied is then:

$$F = \sum_{i=0}^{p} (\mathbf{f}_i^{n\text{-contact}} + \mathbf{f}_i^{T\text{-contact}}), \tag{3.10}$$

where $\mathbf{p}$ is the number of all phalanges being in contact with the virtual object.

3.5.6 Auto Adjustment of γ

Adjusting γ is important to ensure that the user can bring up enough force even for small or heavy objects. This can be done manually with a static value as we have discussed before. However, we propose a way to automatically adapt this value to the physical properties of the virtual object considering its mass, volume, and surface material. Given an object with mass $\mathbf{m}$, a certain gravity acceleration $\mathbf{g}$, e.g., $9,832 m/s^2$ on earth, and its size approximated by a bounding sphere with diameter $\mathbf{d}$, the gravitational acceleration force that we have to overcome is:

$$\mathbf{f}_g = m \cdot g.$$

On the object we can exert a maximum tangential friction force $\mathbf{f}_t \leq \mu \mathbf{f}_n$, with $\mathbf{f}_n = \mathbf{f}^{\text{contact}} \cdot \mathbf{n}$ being the contact force in normal direction, to counteract gravity, such that

$$mg = \mathbf{f}_t \leq \mu \mathbf{f}_n$$

Further

$$mg \leq \mu \mathbf{f}^{\text{contact}} \cdot \mathbf{n}$$

and by plugging in Eq. 3.4 we get

$$mg \leq \mu\gamma(\mathbf{C}_i - \mathbf{p}_i) \cdot \mathbf{n}$$

By assuming a maximum interpenetration depth of 20% of the object bounding sphere in normal direction to the surface, thus

$$\|(\mathbf{C}_i - \mathbf{p}_i) \cdot \mathbf{n}\| \leq 0.2d$$

We obtain

$$mg \leq \mu\gamma 0.2d$$

and Finally,

$$\gamma \geq \frac{mg}{\mu 0.2d} \tag{3.11}$$

The auto adjustment of γ yields a value of approximately 409.1 for most of the objects used in our experiments. With respect to the manual configuration of 400, the result from the auto adjustment comes close to what we would expect for our experiments.

3.6 Visualization of the Hand Model

For the visual representation of the hand in *VR* environments, we use a semi-opaque stylized rigged hand mesh shown in Fig. 3.17. As pointed out by [31], a semi-opaque property helps users to receive a better depth perception and thus seems to support interaction planning. We have noticed that a stylized hand model has the benefit of avoiding uncomfortable user experiences, which can occur with too realistic hand models [50]. The alignment of the physical hand model with the graphics hand mesh is shown in Fig. 3.18.

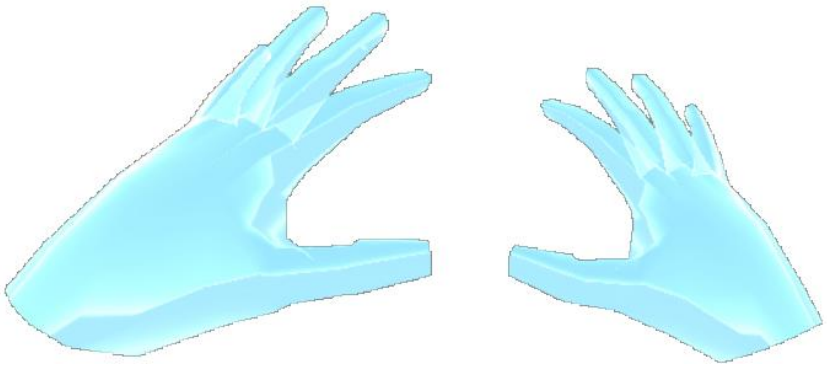


Figure 3.17: Render of the hand mesh.

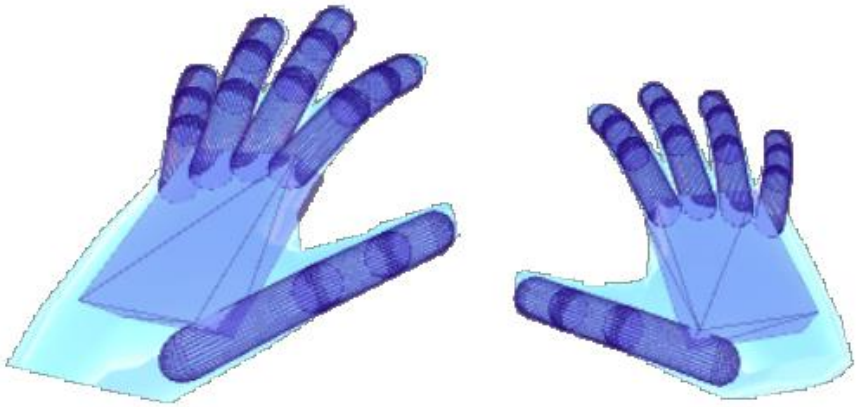


Figure 3.18: Physical hand model aligned with the rendered hand mesh.

Since our tracked hand is able to interpenetrate the virtual object, we render an opaque hand mesh with the pose from the moment an interaction occurred. The hand mesh can still be updated and slightly refined once other fingers interact with the object. This ensures that the user can see a rendered hand pose of its interaction which is not penetrating or causing any confusing visuals. As shown in Fig. 3.19, this should help the user to constantly have a realistic render of the interacting hand pose and avoid visual interpenetration.

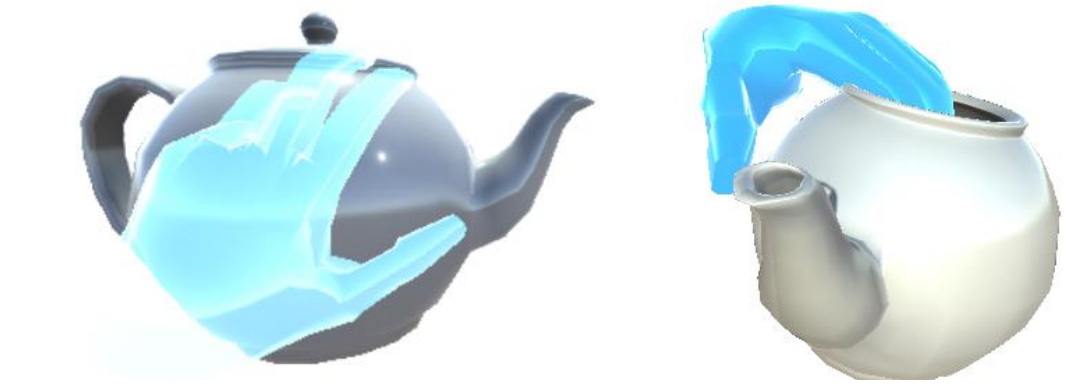


Figure 3.19: Left: Semi-transparent hand mesh rendered before contact. Right: Opaque hand mesh rendered during contact.

For AR environments we render the tracked hand skeleton as an occlusion mask. This makes sense because the user can actually see his real hands in AR and thus it would feel very uncomfortable rendering a second hand mesh. It destroys the perception of using your actual real hand.

The occlusion mask also permits the user to get an actual depth perception of the tracked real hand in relation to the virtual object's 3D position. This means, without the occlusion mask the virtual objects would constantly appear in front of the real hands even if they should be technically occluded with our real hand since no depth test would be performed.

Figure 3.20 has been taken with the HoloLens and shows the occlusion mask where a user is occluding parts of a virtual hologram with his real thumb.

Figure 3.20: Left: The real thumb of the user is occluding a block from the Jenga hologram and thus gives a correct depth perception. Right: Virtual Jenga tower.

During a contact, the visual hand mesh is not allowed to move freely anymore since it would likely interpenetrate the virtual object from a visual point of view. To move the hand mesh still, we have used a simple solution using the global transformation matrix of the virtual object and transfer its rotation and translation onto the hand mesh, assuming a stable contact.

$$\mathbf{T}_{\text{HandMesh}} = \mathbf{T}^{\text{inverse}}_{\text{HandMesh}} \cdot \mathbf{T}_{\text{VirtualObject}}$$

Sound Feedback. In addition to the adaptive mesh rendering in *VR* we have added a subtle sound feedback as soon as a contact between the hand and a virtual object occurs. The sound feedback was also usable in *AR* with the HoloLens. We noticed an improved overall experience due to the subtle sound response from the system.

3.7 Implementation Aspects

We implemented our approach in Unity 5 with PhysX 3. We provide several important implementation details which are critical for efficient integration with the physics engine. These include converting the Leap Motion coordinate space into the world space of Unity 5; finding the contact points between the hands and virtual objects; updating surface points;

applying the forces within the graphics engine; handling arbitrary, possibly non-convex, object shapes; and correctly handling the interpenetration of hand and object.

Hand Model Interpenetration. To avoid interpenetration between the hand model used for visualization and the virtual object, we disable the rigid body property of the corresponding bone once we have estimated a contact point. In Unity, physics colliders that are set as *trigger* do not cause real collision detection nor collision resolution. By setting the corresponding bones as triggers, we prevent the engine to generate any physical reactions between hand bones and the virtual objects, but we are still fully capable to track the current 3D transformation of the hand bones. Fig. 3.21 illustrates this approach on a grasp interaction.

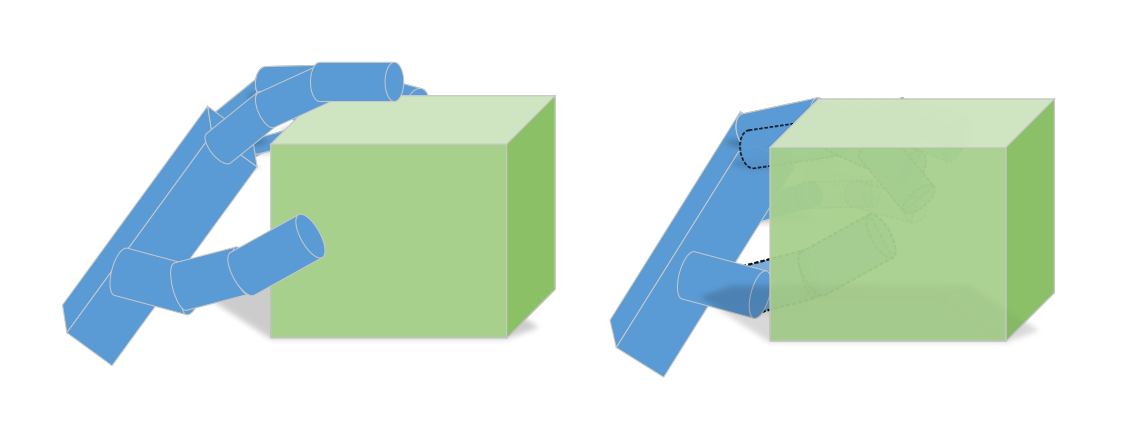


Figure 3.21: Left: Hand pose before interaction. We use rigid bones with all physical properties enabled. Right: Hand pose during interaction. We show the allowed interpenetration with bones set as triggers thus deactivating all physical properties of the intersecting bones.

Contact Point Estimation. Our solution to efficiently identify accurate contact points uses the collision detection of Unity 5. Even if there is no real collision between the hand model and the virtual objects in practice, we are able to estimate accurate contact points: We define a very small Default Contact Offset (DCO), a property of the physics engine, as a threshold to fire a collision event before the actual contact happens as explained in Section 3.5.2. The *DCO* acts as a margin at which point the collision detection fires collision events. At this stage we can identify the specific collision collider of the virtual object. To estimate an accurate surface point on the virtual object, we take the point in $\mathbb{R}^3$ from the collision event within the *DCO* and cast a ray onto the center of the identified sub-collider of the virtual object. The first hit on the collider then represents the estimated contact point. The *DCO* is a rather small area, just large enough to avoid real physical collisions between the hand bones and the virtual objects but small enough

to ensure reasonable contact points via ray casting. We set the collision detection mode of Unity to *Continuous Dynamic* to support fast hand movements.

Surface Point Updates. As long as the contact force remains in the static state indicating a stable interaction the contact points on the object's surface must not be updated on the surface. When the state during contact switches to dynamic, we update the corresponding contact point along the object's surface as illustrated in Fig. 3.8. From the first estimated contact point we apply a transformation as an offset in normal direction of the surface and cast an inverse ray in direction of the new bone position being in dynamic contact state. The ray cast ensures that we can estimate a new contact point directly on arbitrary object surfaces. Our solution detects occluding surface parts of the object by comparing the orientation of the previous normal vector with the normal orientation of the newly proposed surface point. This ensures that a contact remains in a blocked area illustrated with R_4, R_5 in Fig. 3.8. Our proposed solution allows for efficient surface point updates without the need for additional *God-Objects* for every phalanx on the hand skeleton but allows the hand to slide along the objects surface during dynamic contacts.

Applying Forces. Our approach acts as a middleware between the hand input and the physics engine (Nvidia PhysX). We compute the contact force of each bone for every updated physics frame and apply them via the *API (Application Programming Interface)* of the physics engine. The method *AddForceAtPosition(Vector3 force, Vector3 position, ForceMode mode)* allows to explicitly apply forces at specific contact points of any rigid body within the simulation. The first parameter is a Vector3 with magnitude and direction, the second parameter is the actual contact point on the object's surface and the third parameter describes how the forces should be applied, i.e., in which time interval. Choosing *Force* as the desired *ForceMode* allows for evenly exerting the force in Newton (N) across the period of frame updates. High accelerating forces such as explosion can be modeled with *Impulse* as *ForceMode* which is applied in Newton-Seconds (Ns).

Arbitrary Object Shapes. Physics engines have problems with complex, non-convex object meshes and only support convex objects or primitive shapes [9, 45]. Thus, Nvidia PhysX 3 does not support collisions between concave mesh colliders and dynamic non-kinematic rigid bodies. A common solution is to approximate a non-convex shape by its convex hull [45]. However, convex mesh approximations are too imprecise representations of the object mesh and this would significantly limit the interaction possibilities for fine-grained dexterous manipulation. Therefore, we approximate the concave objects with small voxels, by using a Finite Element Method [49]. Figure 3.22 - Fig. 3.25 show the discrete voxel approximation with varying Level of Detail (LoD).

Figure 3.22: Physical object representation: very high *LoD*

Figure 3.23: Physical object representation: high *LoD*

Figure 3.24: Physical object representation: medium *LoD*

Figure 3.25: Physical object representation: low *LoD*

External Forces. In our implementation, the external forces such as gravitational acceleration and collision forces between objects are handled by the physics engine internally. Since we handle the entire physics interaction between the hand and virtual objects ourselves, all applied forces influence and counteract each other. Thus, an unstable grasp midair would cause a virtual object to fall down because the applied contact forces were not sufficient to counteract gravity.

Collision forces between objects are also simulated by the physics engine which can however influence computation of the contact force.

3.8 Additional Environment Details

This section covers some relevant details about the periphery and hardware which are not directly related to the core of our method but rather belong to the development environment. We discuss how to correctly transform the tracking data from the Leap Motion device; how to use an image pass-through of the Leap Motion with a potential Head Mounted Display (HMD); illustrate the HoloLens tracking to synchronize the virtual render camera with the real 3D position in the room; and show the spatial mapping of the HoloLens which enables us to place and attach virtual objects in reality.

3.8.1 Converting Leap Motion Data

In Section 3.3.2 we have pointed out the transformation accounting for the physical mount correction of the Leap Motion device to match the hand perception with our biological eyes. To bring the Leap Motion data sufficiently into the world space of a graphics engine there have to be done several conversions in addition to that.

The Leap Motion's configuration is set as up-facing device by default, called the desktop mode. To use the device in a front-facing *HMD* mode, it is mandatory to apply the following transformation which represents a 90 degree rotation around the x-axis and a 180 degree rotation around the z-axis as long as the physical mount is square to the *HMD* projecting the y-axis in forward direction:

$$\mathbf{M}_{\text{Desktop}\mapsto\text{HMD}} = \mathbf{T}_{\text{Mount}} \cdot \begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Leap Motion uses units of millimeter, where other graphics engines may use intern

units of meters. Thus, it is necessary to convert the coordinates derived from the Leap Motion by applying a scale of 0.001 to the tracking data. This changes the linear unit of measurement as the following matrix describes:

$$\mathbf{S}_{\text{mm}\mapsto\text{m}} = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The next step accounts for the conversion from Leap Motion's right-handed coordinate system to a left-handed coordinate system of the graphics engine. This is achieved by the following matrix which represents a negative scaling of the z-axis by -1:

$$\mathbf{S}_{\text{righthand}\mapsto\text{lefthand}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As a last step one has to consider the transformation derived from the *HMD* in $\mathbb{R}^3$. The mapping of the tracking data can be achieved by multiplying the matrices above as:

$$\mathbf{M}_{\text{HMD}} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{T}_{\text{Mount}} \cdot \begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

However, the discussed transformations do not account for any other eventual physical rotations. For tilted mounts, the following matrices are important to take arbitrary rotations into account where α, β and γ represent the corresponding rotations around the x, y and z-axis:

$$\mathbf{R}_{\mathrm{x}} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & \cos\alpha & -\sin\alpha & 1 \\ 0 & \sin\alpha & \cos\alpha & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{R}_{\mathrm{y}} = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 1 \\ 0 & 1 & 0 & 1 \\ -\sin\beta & 0 & \cos\beta & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_{\mathrm{z}} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 1 \\ \sin\gamma & \cos\gamma & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

All three matrices can be combined into one rotation matrix, describing a tilted mount via multiplication:

$$\mathbf{R}_{\mathrm{Mount}} = \mathbf{R}_{\mathrm{x}} \cdot \mathbf{R}_{\mathrm{y}} \cdot \mathbf{R}_{\mathrm{z}}$$

Thus, $\mathbf{T}_{\mathrm{Desktop}\mapsto\mathrm{HMD}}$ has to be changed by multiplying $\mathbf{T}_{\mathrm{Mount}}$ accounting for the translation with $\mathbf{R}_{\mathrm{Mount}}$ for the physical device rotation:

$$\mathbf{M}_{\mathrm{Desktop}\mapsto\mathrm{HMD}} = \mathbf{T}_{\mathrm{Mount}} \cdot \mathbf{R}_{\mathrm{Mount}}$$

In *VR* and even more so in *AR* the user can move within the real world while the $\mathbb{R}^3$ transformation inside the world space is constantly being tracked and mapped to the virtual computer graphics world space. Thus it is necessary to align the tracking data of the hand skeletons with the virtual camera setup of the graphics engine. A very convenient way to do so in Unity 5 is to attach the Leap Motion controller as a Game Object (GO) to the virtual camera *GO* and applying the corresponding mount corrections. Figure 3.26 illustrate the relative transformation between the Leap Motion *GO* and the virtual camera *GO*. With Unity 5 this can be achieved by attaching the Leap Motion *GO* with the virtual (mono) camera *GO*. This ensures that the camera transformations which are obtained from the head tracking in $\mathbb{R}^3$ can directly be applied to the Leap Motion

GO due to the hierarchical *Child-Parent Concept* of Unity 5.

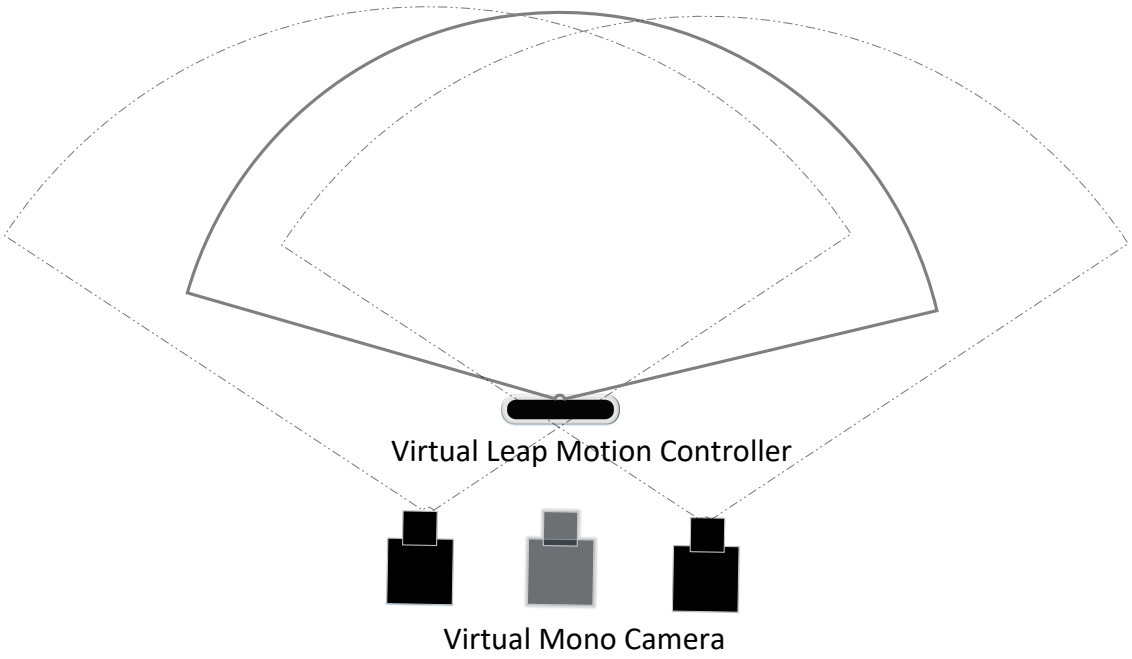


Figure 3.26: Hierarchical *GO* structure of virtual render camera and the virtual motion controller. The controller is attached to the camera as a child object including the transformation accounting for the physical mount correction. The hierarchical structure allowing for relative transformations from the parent *GO*.

It is important to point out how the transformation process of the tracking data looks like to consider different development environments. However, these steps are provided by Leap Motion's *Core Assets* for Unity 5.

3.8.2 Image Pass-Through Stereo Alignment

This section covers the process of aligning Leap Motion hands with real hands for *AR* applications using the physical stereo cameras of the Leap Motion device. The overall alignment problem originates from the different baseline of human eyes and the virtual cameras on the Leap Motion device which differ in the perception of scale. However, this is only a problem if the Leap Motion cameras are used as image pass-through for *AR*. Using the Leap Motion's physical cameras as image pass-through enables *AR* experiences even without *OSTMHD* such as the Oculus Rift or HTC Vive. Since we were using the HoloLens which is an *OSTMHD* this problem did not occur for us because we could use our very own biological eyes to capture the real world.

The perception of world scale derived from different stereo setups featuring different baselines will show deviations. The Interpupillary Distance (IPD) of humans is approximately 63mm [13] considering the mean values of both male and female test persons. where the Inter Camera Distance (ICD) of the Leap Motion accounts for 40mm. Figure 3.27 illustrates the difference between *IPD* and Leap Motion's *ICD*. The different baseline gives a stereo disparity when using Leap Motion as image pass-through.

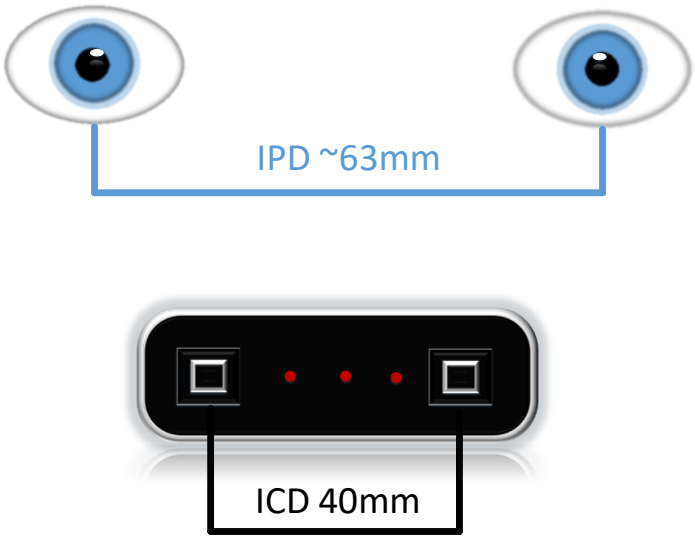


Figure 3.27: Difference between *IPD* and *ICD*.

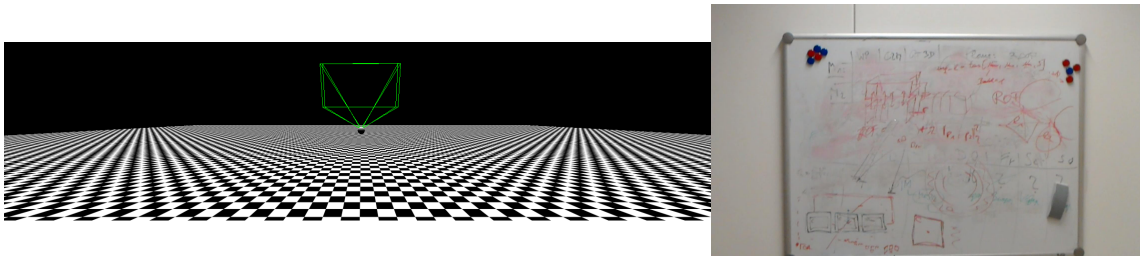
The problem can be solved by matching the Virtual Camera Distance (VCD) with the *ICD* instead of the *IPD*. Thus, the real world scale perception of the *ICD* is matched with the *VCD* from the Leap Motion device when using image pass-through with *HMD*s.

3.8.3 HoloLens - Tracking

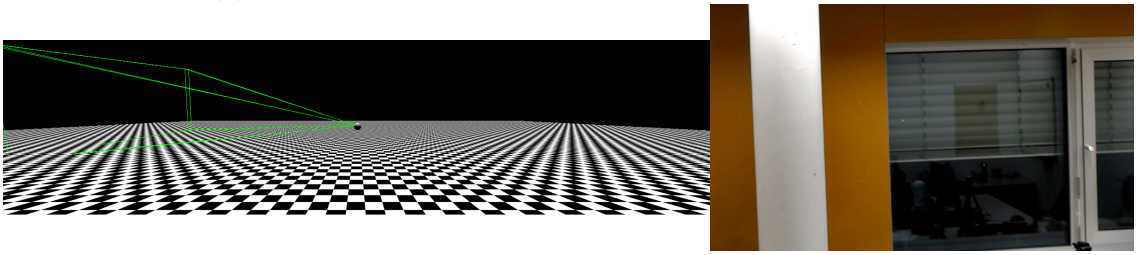
For *AR* scenes it is mandatory to constantly track and map the HoloLens' 3D position and rotation in reality as we need to synchronize the physical camera transformation on the device with the virtual render camera. Such algorithms are called Simultaneous Localization and Mapping (SLAM) systems. For the integration of a monocluar *SLAM* within a real-time graphics engine written in DirectX or OpenGL we refer to [19].

Using the HoloLens SDK this part together with the stereo rendering is straight forward. Figure 3.28 shows various camera orientations of the HoloLens in the real room. The physical camera is facing in forward direction in (a), rotated around the Y-axis to the

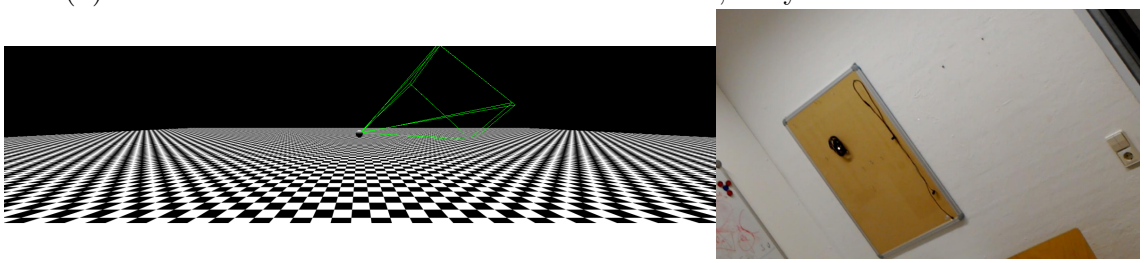
left in (b) and rotated slightly to the right around the Y-axis and Z-axis in (c).



(a) HoloLens forward orientation facing our whiteboard.



(b) HoloLens rotation around the Y-axis to the left, away from the whiteboard.

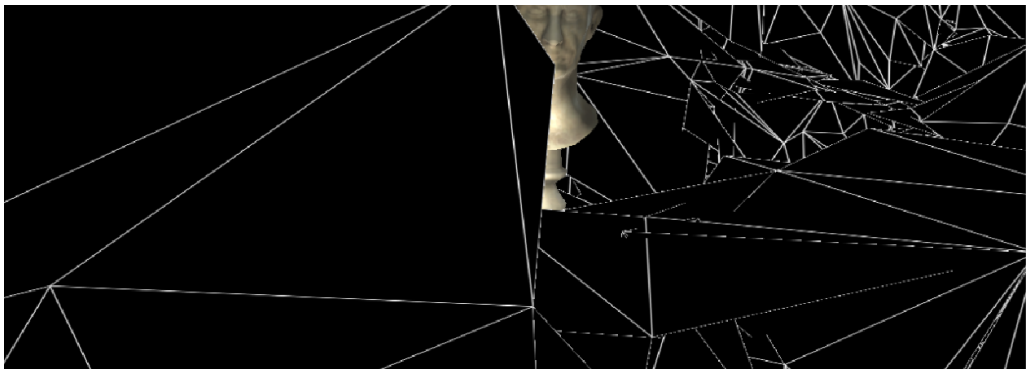


(b) HoloLens rotation around the Y-axis and Z-axis to the right with different position. The whiteboard is only slightly visible in the corner.

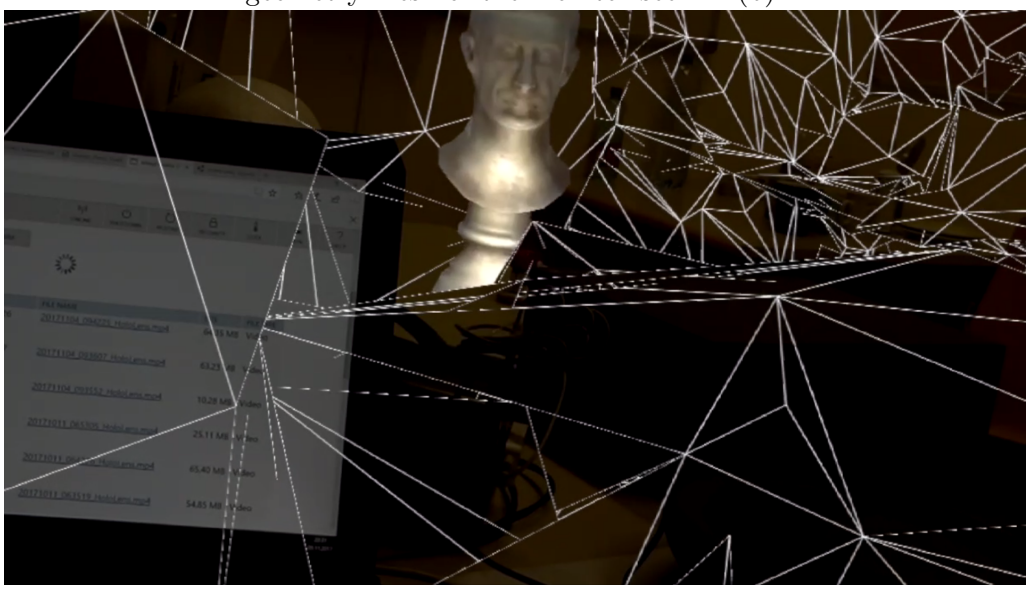
Figure 3.28: This figure illustrates the tracking of the HoloLens camera inside the real world. This tracking information is synchronized with the virtual render camera of the graphics engine for *AR* experiences. Left: Virtual camera tracking. Right: Real HoloLens camera position and rotation.

3.8.4 HoloLens - Spatial Mapping

The HoloLens provides us with a geometric surface description of the real world represented by a triangle mesh. This mesh is used to create convincing *AR* experiences. The mesh is constructed and constantly refined during runtime. We can use it to gather actual geometric understanding of the real world and place virtual objects on such geometry, e.g., on a real desk. Thus, it is possible to let the virtual objects collide with geometry from reality. Geometry from the real world can also occlude virtual objects which is shown in Fig. 3.29.



(a) Spatial Mapping shown with only the virtual geometry. The figure is occluded by the geometry mesh of the monitor seen in (b).



(b) Spatial Mapping through the HoloLens from a different point of view. The figurine is occluded by the geometry mesh of a box.

Figure 3.29: These images show the Spatial Mapping, once with only the virtual geometry in (a) and once with the camera stream in (b) from a different angle. Note: the world mesh may show distortions which arised from recording issues with the HoloLens.

With the geometric description of the real world, we are able to attach virtual objects directly in the real world to interact with them. The 3D position of virtual objects in relation to the constructed geometric mesh of the real world can be saved in form of *World Anchors* which simply hold exact information of the object's 3D transformation with respect to the internal model of the real world.

3.9 Kinematic Approach

In this section we shortly reflect our first simple hand-object interaction approach. Before we came up with our novel analytic model we have attempted in-depth research of recent related work as discussed in Chapter 2. What we had in mind first was to try a simple method just to grab virtual figurines in chessboard environments using defined collision states. Thus, we decided to start with a kinematic method which directly transfers translations and rotations in $\mathbb{R}^3$, derived from the global hand motion onto the virtual objects. To express the object motion we have used the simple kinematic object transformation mentioned in [23]:

$$\mathbf{M}_{\text{object}} = \mathbf{M}^{\text{hand}}_{\text{object}} \cdot \mathbf{M}_{\text{hand}},$$

which transfers the global transformation of the hand skeleton directly onto the virtual object. With this method we could already achieve some first results, grasp objects and place them on different positions.

However, this method was still very limited and does not cover the capabilities of the human hand for realistic object manipulations. The kinematic method was a very good starting point to get familiar with the recent state of related work while giving us knowledge about current limitations. After the kinematic prototype we have started to design our introduced novel approach for direct object manipulations which overcomes current limitations.

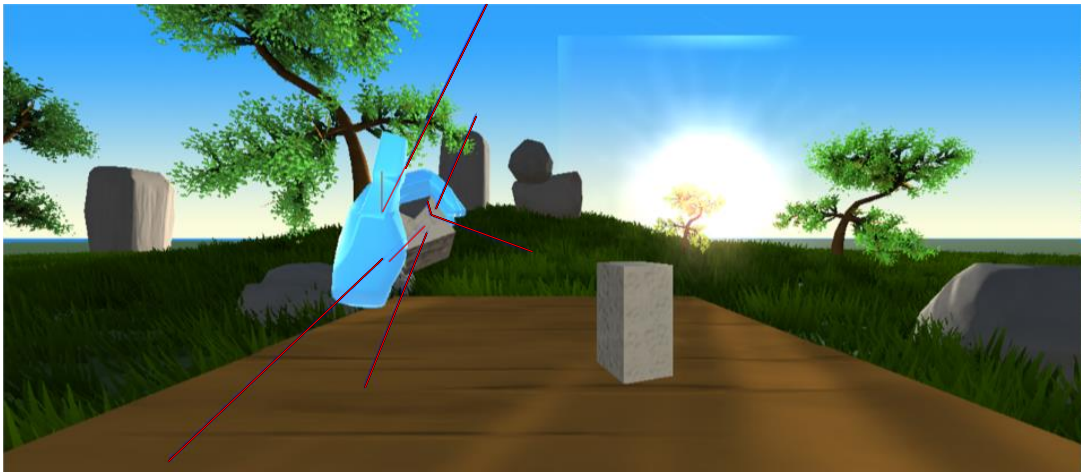
4 Results

We present here several interactions that demonstrate the capabilities of our approach. During all our tests, our approach could manage stable frame-rates without any performance issues. For the Virtual Reality (VR) environments we have added complex shaders, foliage, transparency, lighting, etc. to stress the graphics engine. Still, the test scenes run at over 60fps on an Intel i7 with 2.6GHz with an NVIDIA GTX 980M graphics card.

Every experiment indicates the exerted contact forces depicted as red arrows. For the sake of demonstration we visualize only the contact forces according to each estimated contact point instead of the whole contact patch.

Fig. 4.1 shows a scene where the user grasps different objects. The user grasps and lifts a cube in his/her hand (a). Then, the user grasps an axe, with his/her fingers and palm to further use the axe to hit a tower with cubes that subsequently collapses (b). Finally, the user interacts with a small figurine, which is lifted by a flat hand.

Fig. 4.2 (a) shows a user catching a block with the left hand from a tower which had been knocked over by the right physics hand. Then the user is handing over a virtual object midair to a second physics hand in (b). This shows that our method naturally supports multi-hand interactions like in [24]. Since our approach is purely physics-based, another virtual hand simply represents another medium which exerts forces on an object. That is why our method supports multi-hand interactions simply by design. In (c) the user is balancing an object midair with the left fingers and a greater amount of contact force induced by the thumb of the right hand.

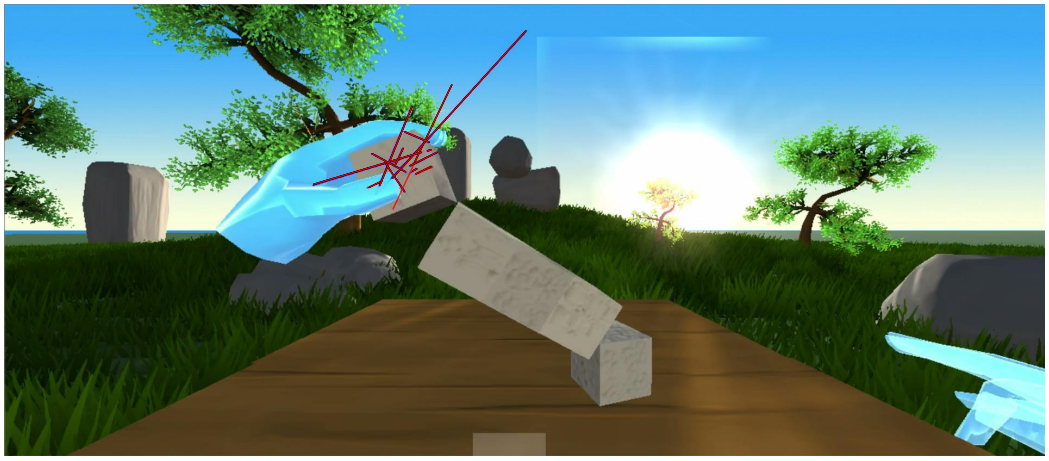


(a) Palmar grasping.

(b) Grasp an object with fingers and use it as a tool on other objects.

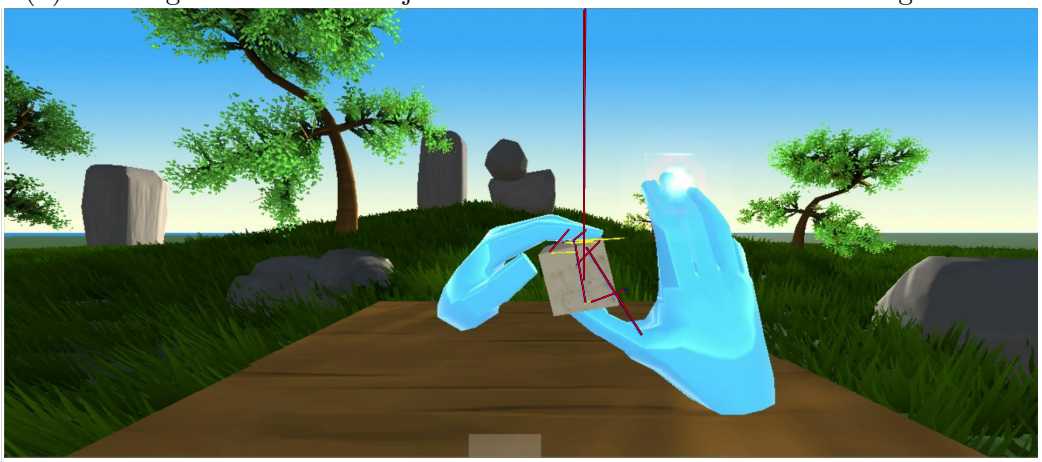
(c) Lift an object by using only a flat hand pose.

Figure 4.1: Different interaction results. We show the contact forces in red, and illustrate the global hand motion in yellow. Please see text for more details about the scenes.



(a) Grasping a cube midair from a tower of cubes in motion.

(b) Handing over a virtual object midair from the left hand to the right hand.

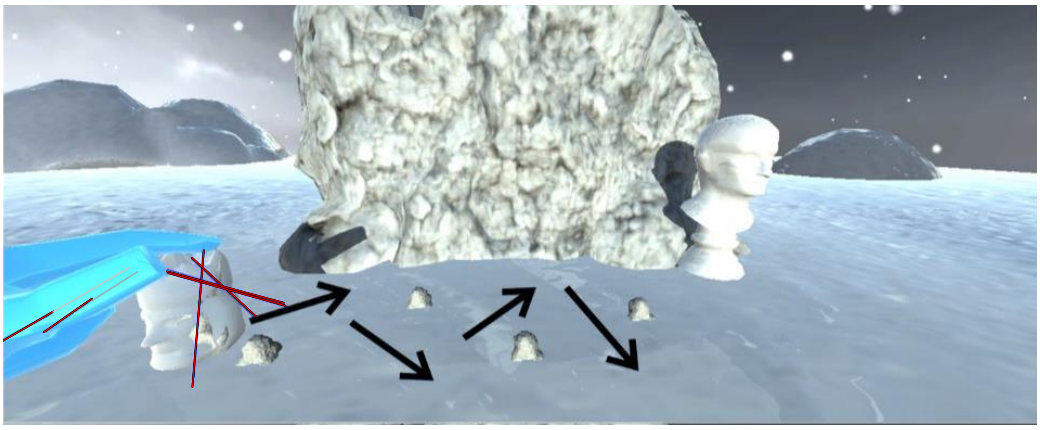


(c) Unconstrained interaction pose for balancing an object midair using both hands.

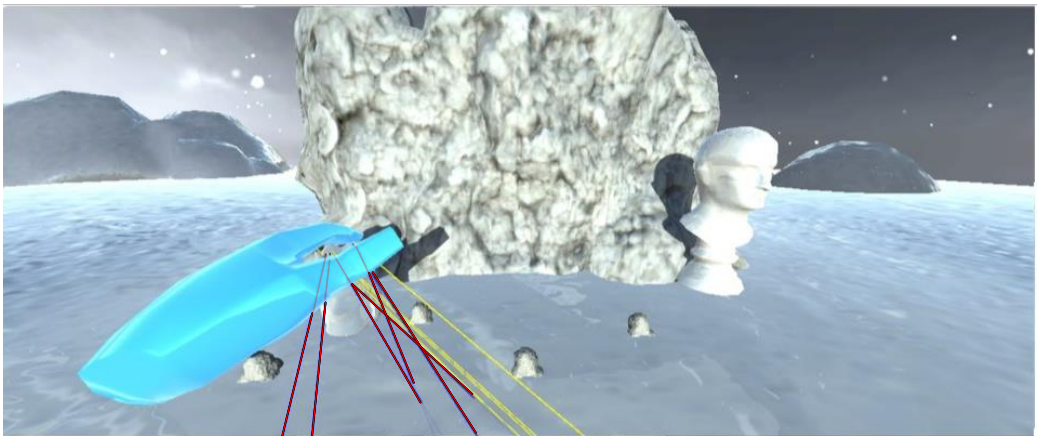
Figure 4.2: Balancing objects midair. We show the contact forces in red, and illustrate the global hand motion in yellow. Please see text for more details about the scenes.

Fig. 4.3 shows a user sliding a cylindrical head sculpture along a path, depicted by arrows, by pushing the object on a slippery ground. Therefore, we use a large friction coefficient between the hand and the object, and a small friction coefficient between the object and the ground. First it is shown in image (a) how a user is inducing contact force to prepare a stable contact and thus a higher friction portion. Then the user additionally induces higher tangential forces due to the global hand motion to really make the object slide along the path. As shown in the next image, the figurine can finely be moved along the indicated path by inducing forces like in reality. Performing the exact same manipulation but with a higher friction coefficient (e.g., wooden material) for the ground the exerted forces would induce the object to tilt instead of slide on the ground.

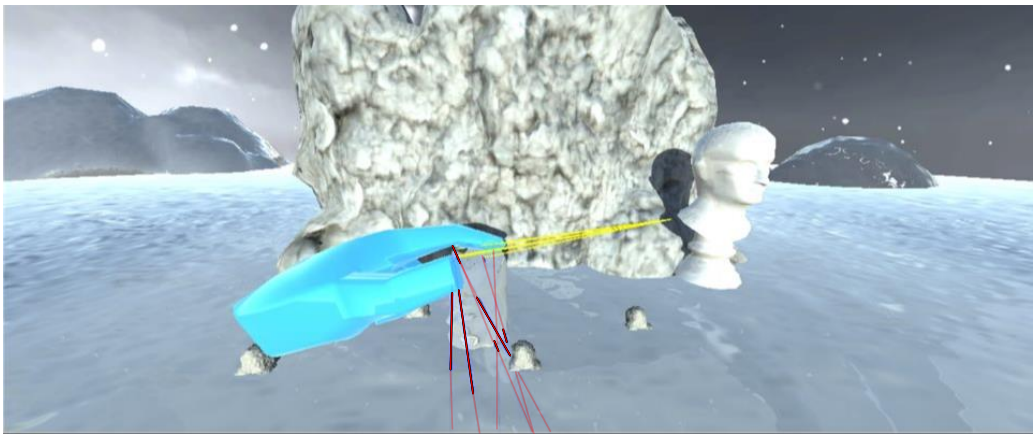
Fig. 4.4 shows that our approach is capable of stable motion control. In reality a grasped object is controlled by the motion of the grasping hand due to the involved forces. We demonstrate a stable motion control midair starting from an initial grasping position in image (a). We also show here the voxelization of the concave shaped figurine object explained in Section 3.7. In image (b), the grasped hand is translated and rotated towards the render camera. We show that the grasped figurine object is correctly moved as well only by inducing the forces from the grasping hand. Then, the hand is rotated in the other direction, away from the virtual camera and the object is again correctly moved by only inducing the involved forces.



(a) Inducing contact force on a figurine to increase the friction between hand and the virtual object to make it slide along with he hand in the following images.

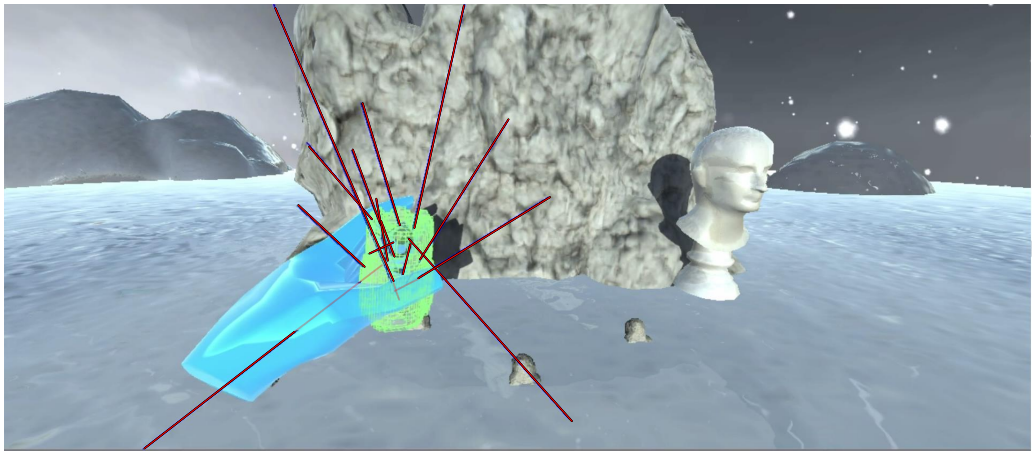


(b) Controlling the figurine's movement along the path by inducing friction forces.

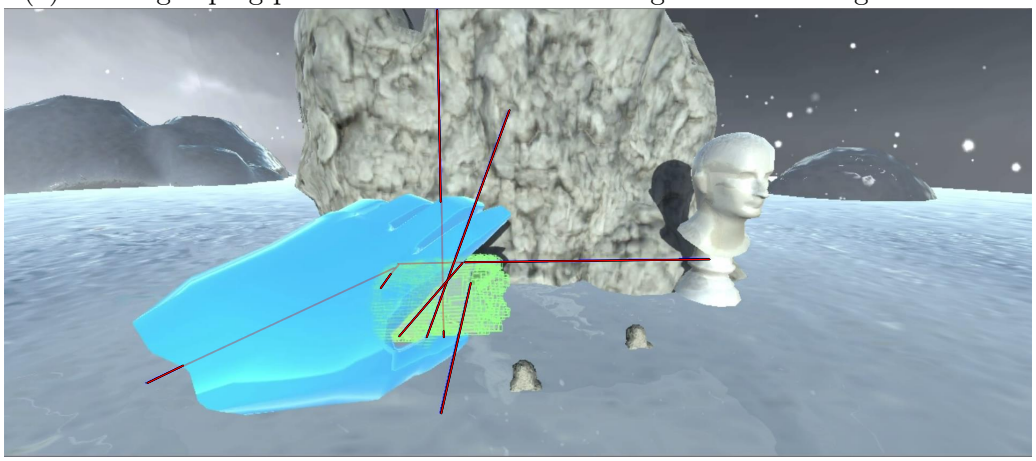


(c) Continuing to make the object slide along the path by using the forces.

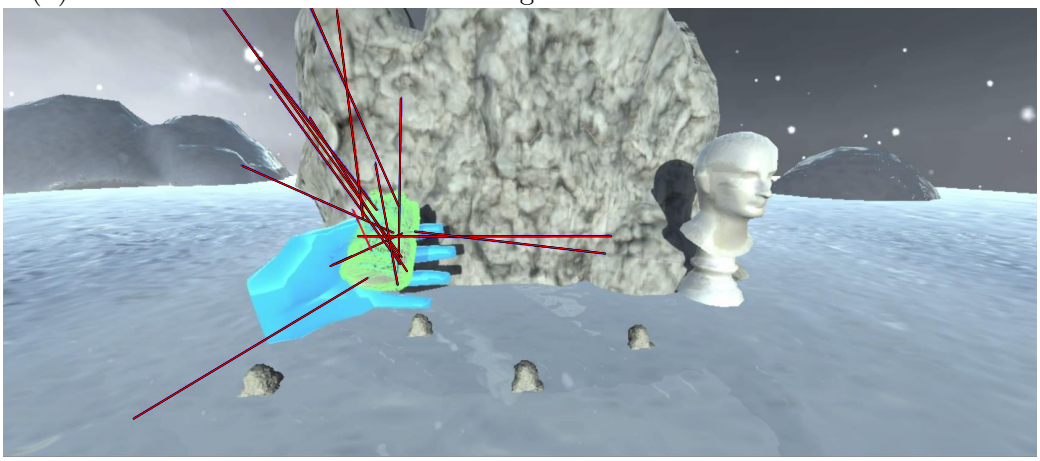
Figure 4.3: This figure shows how a user can control the motion of a figurine object by inducing forces to let it slide along a certain path indicated by the black arrows in the first image. We show the contact forces in red, and illustrate the global hand motion in yellow. Please see text for more details about the scenes.



(a) Initial grasping position midair before rotating and translating the hand.



(b) Stable motion control after rotating the hand towards the virtual camera.



(c) Stable motion control after rotating the grasping hand away from the virtual camera.

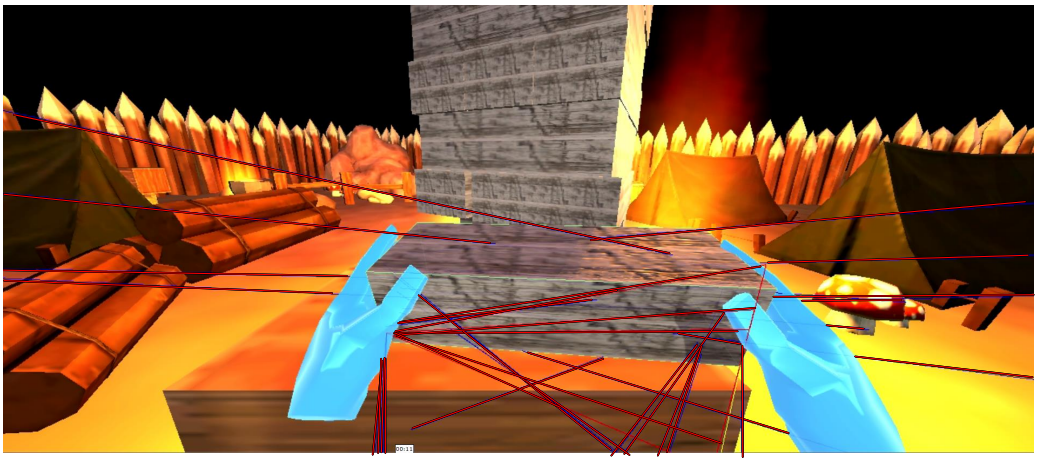
Figure 4.4: This figure shows that our method is capable of stable motion control for stable grasp interactions. This means, if the hand is rotated or moved during grasping, the object is moved correctly as well due to the involved forces. We show the contact forces in red. Please see text for more details about the scenes.

Fig. 4.5 shows a user pulling blocks from a tower using both hands, similar to the game *Jenga*, using only the friction between the fingers and the blocks. The user can use a single hand and both hands to pull, grasp, or lift the blocks. Our interaction method is used to calculate the friction forces between object and hand, whereas the physics engine simulates the interactions between the different blocks.

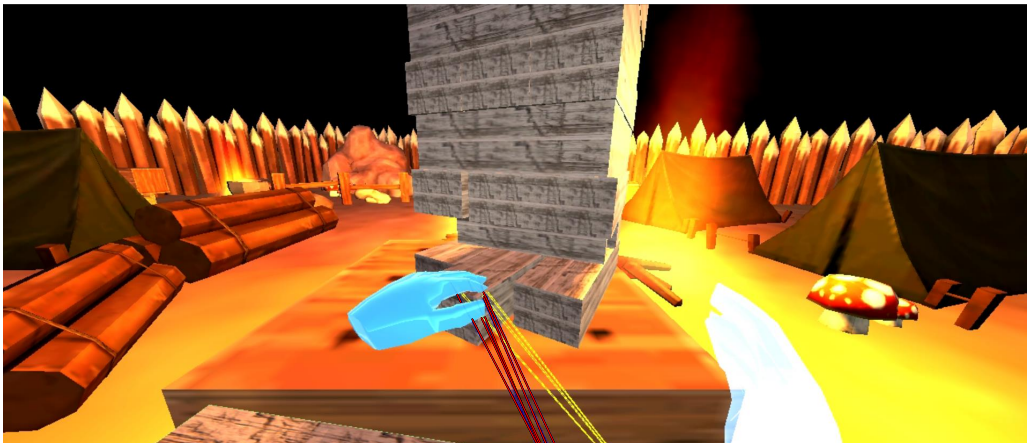
Fig. 4.6 shows object manipulations in Augmented Reality (AR) using the HoloLens. In the first image we show the occlusion mask of both hands as the user is pulling out a block from the tower. We prepared this particular scene for the *OpenLAB-Night 2017* of the Institute of Computer Graphics and Vision (ICG) where users could use our method to pull out as many blocks as possible without causing the tower to collapse. Image (b) shows a grasp of a figurine object with the corresponding occlusion mask.

Fig. 4.7 shows very dexterous object manipulations where the user can use his fingers to finely control and induce the forces. The illustrated examples are even a challenge in reality though our method is still capable of performing such kinds of dexterous hand-object interactions. The left side of the images shows the corresponding hand-object manipulation from reality and the right side shows the same dexterous interaction using our physics-based method.

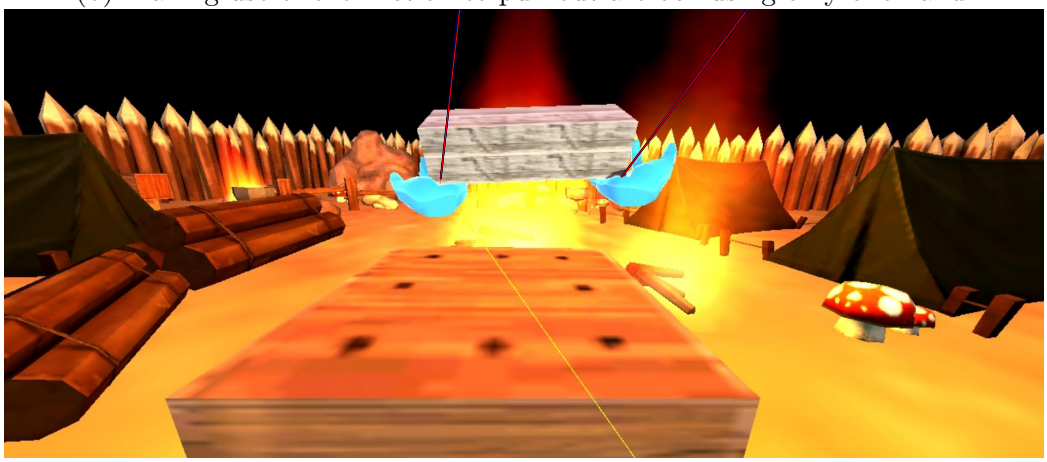
Note that for all of these interactions, there is no kinematic grasping involved, as for example used by [23, 24]. In our approach the grasping is made possible by modeling the friction forces between the object and the hand. That is why our method offers more diverse and realistic object manipulation, covering the complexity of human hand interactions.



(a) Using both hands to pull out a block from a tower.

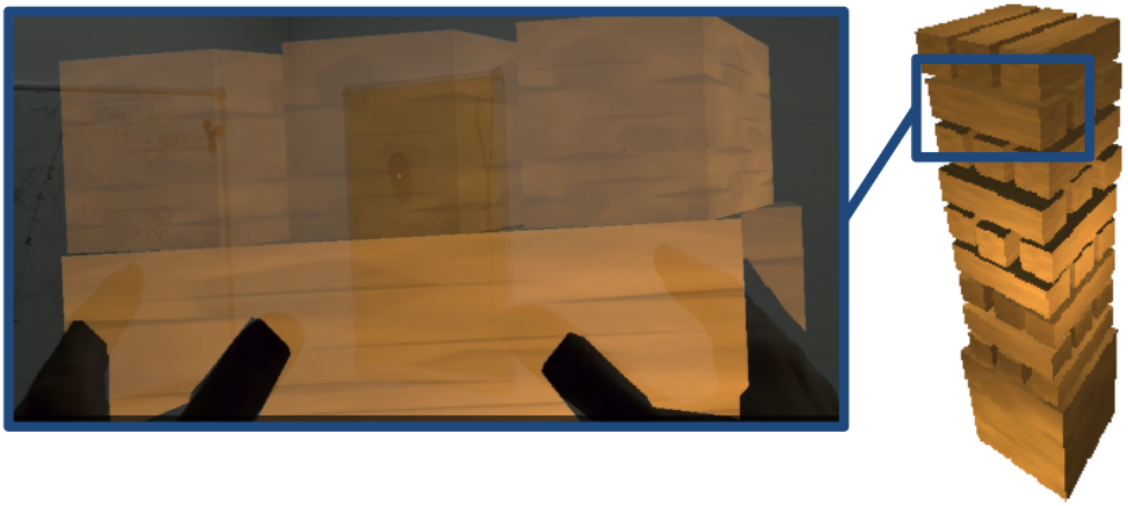


(b) Making use of the friction to pull out a block using only one hand.



(c) Lifting a block with both hands.

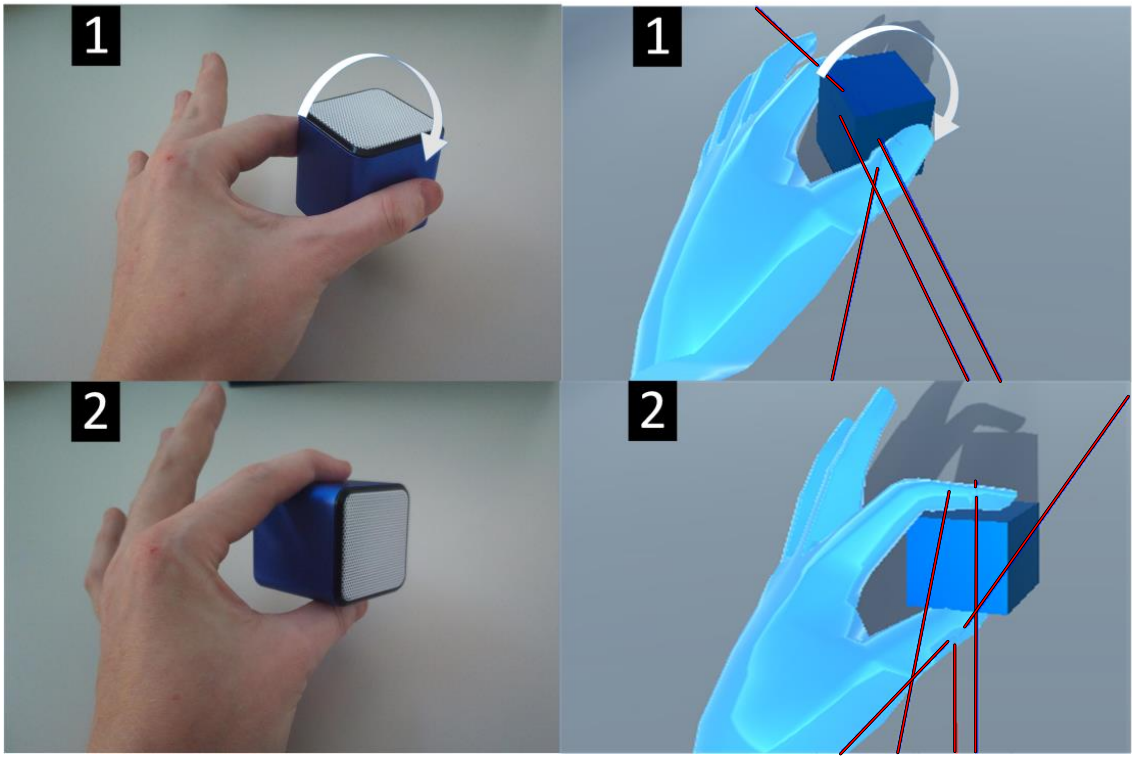
Figure 4.5: This figure shows that our method can be used to carefully pull out blocks of a tower without making the tower collapse. We show the contact forces in red, and illustrate the global hand motion in yellow. Please see text for more details about the scenes.



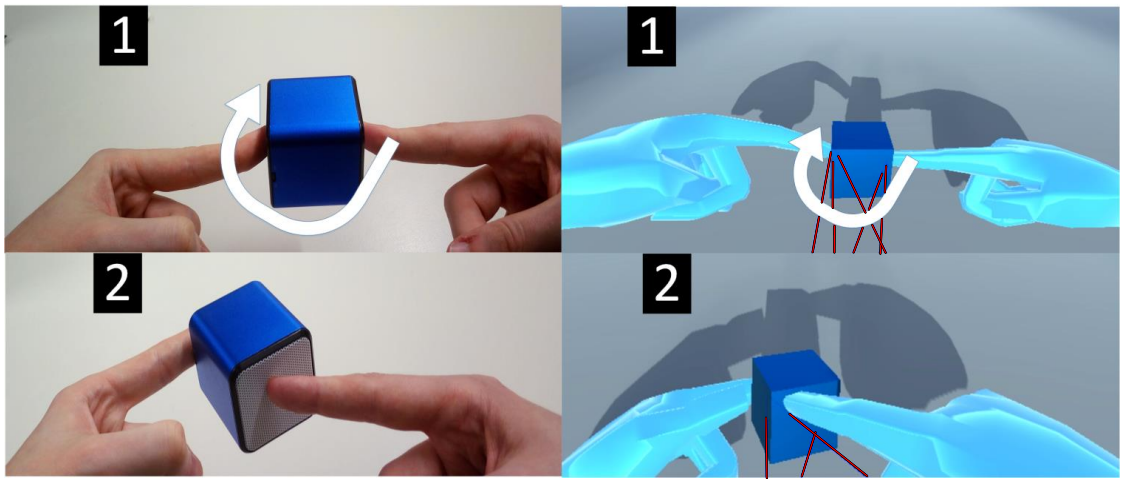
(a) Using both hands to pull out a block from a large *AR* tower.

(b) Grasping a figurine object.

Figure 4.6: This figure shows object interactions in *AR* with occlusion mask. The first image shows a user which pulls out a block of a tower using both hands which is similar to the interactive live demo that we have presented at the *OpenLAB-Night 2017*. The second image shows the occlusion mask during a grasp. These particular images have been made with the HoloLens.



(a) Finely inducing forces with two fingers to spin an object.



(b) Dexterous object manipulation using both index fingers to spin the object from the right side midair.

Figure 4.7: This figure shows that our method supports very dexterous object manipulations which are even hard to perform in reality. On the left side we show how to spin a real object by using fingers. On the right side we perform the same dexterous manipulation using our method. The objects are manipulated by carefully inducing forces. We show the contact forces in red. Please see text for more details about the scenes.

OpenLAB-Night Demo As mentioned previously, we have presented our physics-based approach at the *OpenLAB-Night 2017* of the *ICG* at the *Technical University of Graz*. We prepared two live demos, one for *VR* with the Oculus Rift DK2 and one for *AR* with the HoloLens. For the *VR* demo we have used a similar virtual environment like shown in Fig. 4.1 and Fig. 4.2 where we placed different kinds of virtual objects around the origin position of the user. The users could simply try out our interaction method and use the physics forces to induce certain object manipulations like building a tower from smaller virtual cubes, placing a more complex object on top of it or try using both hands to lift an object.

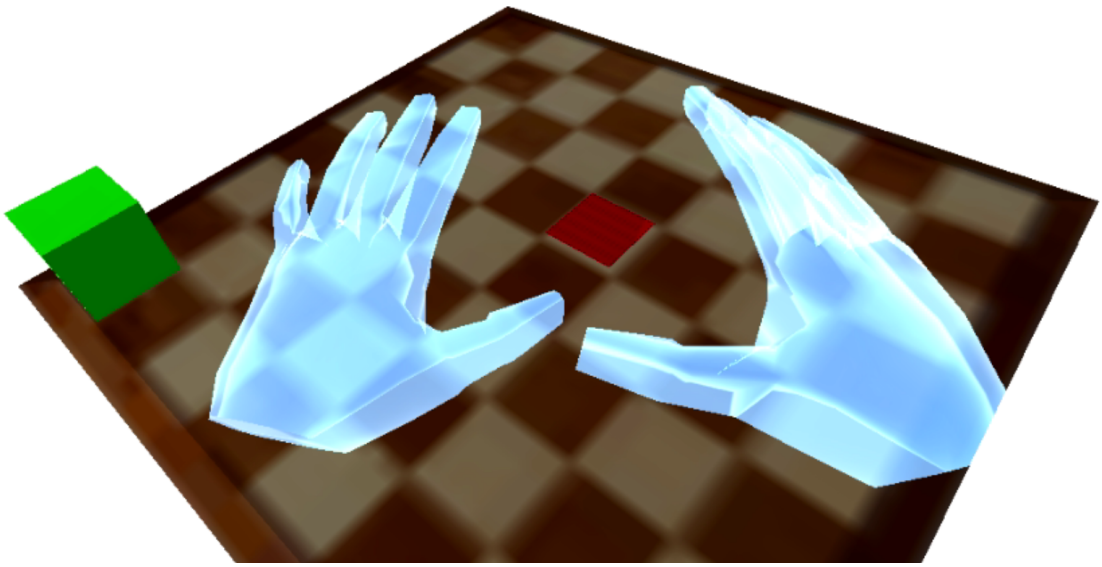
For the *AR* live demo we have prepared a virtual tower as shown in Fig. 4.6 and augmented it as a virtual object on a real table. The guests could use their real hands to pull out as many blocks as possible from the tower. We are very grateful for the motivating feedback which we received from the publicity. It was a pleasure to see how the guests enjoyed to use our physics-based hand-object interaction method while trying out its capabilities.

5 Evaluation

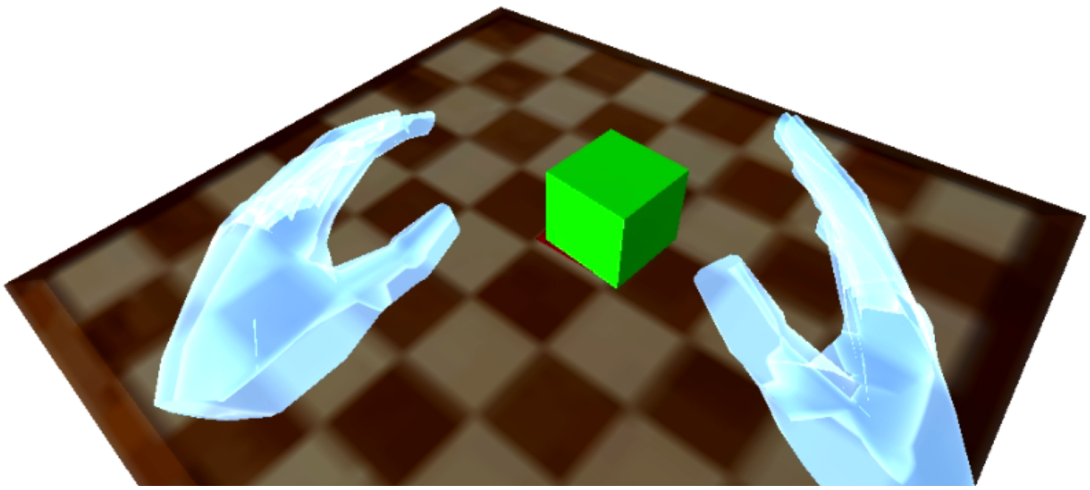
In this section, we evaluate our approach for hand-object interaction. We conducted a pilot study among the users that participated to the evaluation process. Also, we have evaluated our approach in terms of real-time performance which is, together with realistic interaction possibilities, the most critical aspect regarding our contribution. In addition to that, we provide a quantitative evaluation of our approach where we compare against different state-of-the-art interaction methods in terms of positioning accuracy. We created the simple Virtual Reality (VR) test environment shown in Fig. 5.1 where the users first could get familiar with each of the evaluated interaction methods and get a sense of its capabilities. The same scene has been used for the quantitative evaluation. We used the Oculus Rift DK2 to display the 3D scene to the users and the Leap Motion sensor mounted on the front plate. 6 different users participated in our evaluation. All the users were familiar with *VR*/Augmented Reality (AR), and half of the users had hands-on experiences with *VR*/*AR*.

We compare our approach with two other recent approaches. The first baseline is our re-implementation of the kinematic part of [23]. In this approach, the object is considered as grasped when some measured forces reach a self defined threshold. Then the global transformation of the hand is applied to the object which is set to kinematic. We re-implemented the same kinematic motion control, while we simplified the prior grasp condition. As a second baseline we use an industry method, the Leap Motion Interaction Engine 1.0[1], which is also a kinematic approach. We had to use a cubic form for the test object since the used Interaction Engine is strictly limited to simple primitive shapes.

[1]Publicly available at `https://developer.leapmotion.com/unity/`



(a) Starting position as initial configuration.



(b) Successfully completed configuration.

Figure 5.1: Scene used for evaluation. It shows a chessboard with a cube object, and a user is asked to move the cube from a starting position shown in green to a specified location on the chessboard shown in red. The initial configuration is shown in image (a), the configuration after the interaction is completed is shown in image (b).

5.1 Pilot Study

As we conducted the pilot study, the users were given several questions as shown in Table 5.1. Each question can be rated from 1 to 5, 5 being the best grade. 'Naturalness' refers to the question "How natural does the interaction feel?" and 'Diversity' refers to the question "Rate the diversity of interaction possibilities". The exact formulation of all the questions are given below.

1. How natural does the interaction system feel? – "Naturalness"
2. How useful is this interaction system for VR / AR? – "Usefulness"
3. Rate the diversity of interaction possibilities! – "Diversity"
4. How easy is the interaction method to learn? – "Easy learning"
5. Which interaction method did you prefer? – "Preference"

Input method →	**Ours**	Re-impl.	Interaction
User question ↓		[23]	Engine
Naturalness	$\mathbf{4.2 \pm 0.7}$ / **4**	2 ± 1.1 / 2	2.3 ± 1.0 / 3
Usefulness	$\mathbf{4.5 \pm 0.5}$ / **4.5**	1.8 ± 0.4 / 2	3.2 ± 0.7 / 3
Diversity	$\mathbf{4.5 \pm 0.5}$ / **4.5**	2.6 ± 0.5 / 3	1.6 ± 0.8 / 1.5
Easy learning	3.3 ± 1.0 / 3	4 ± 0.9 / 4	$\mathbf{4.6 \pm 0.5}$ / **5**
Preference	**4**	0	2

Table 5.1: Results of our pilot study. We asked the users to answer several questions about the interaction method and rate them from 1 to 5. Users rated our method more useful, more natural, with more diverse interaction possibilities, but more difficult to learn. Still, 4 out of 6 users preferred our physics-based method over the other two methods.

We conclude that our system is much more powerful as it allows for more realistic and diverse interactions once the user is used to it.

Detailed Evaluation of the Pilot Study In addition to the results from the pilot study shown in table 5.1, we present a more detailed assessment of the first four questions in Fig. 5.2 - Fig. 5.5 below.

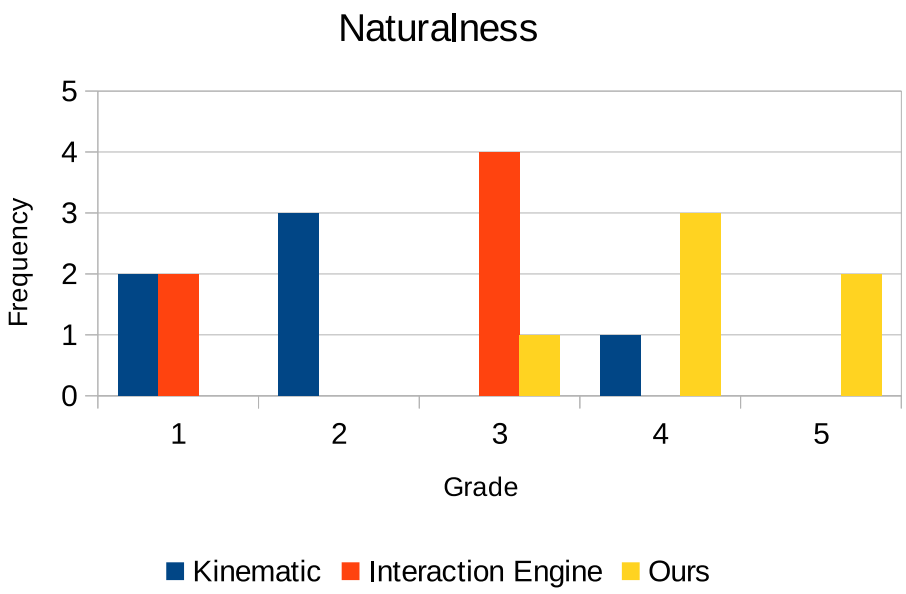


Figure 5.2: Histogram of user answers for the "Naturalness" question.

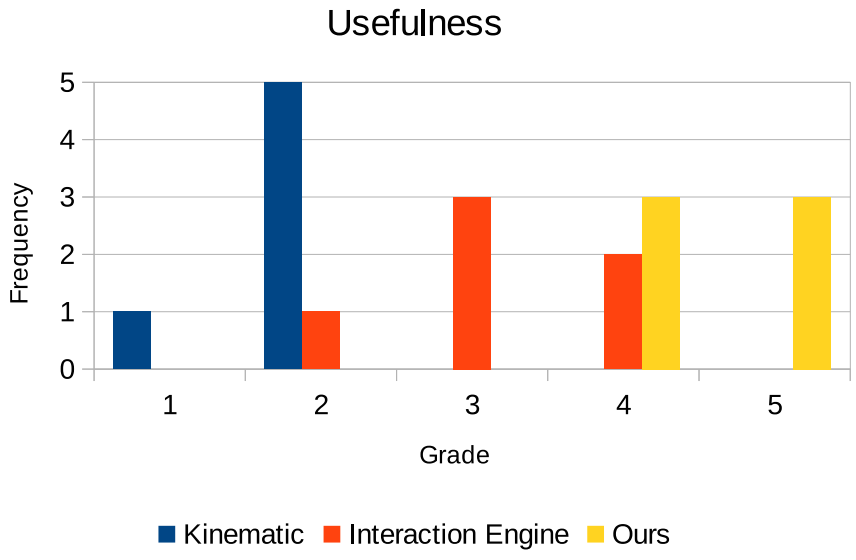


Figure 5.3: Histogram of user answers for the "Usefulness" question.

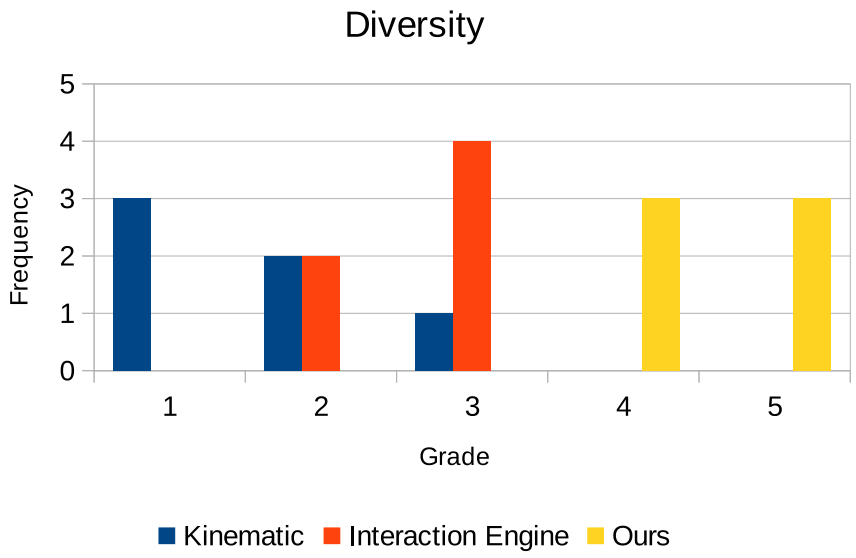


Figure 5.4: Histogram of user answers for the "Diversity" question.

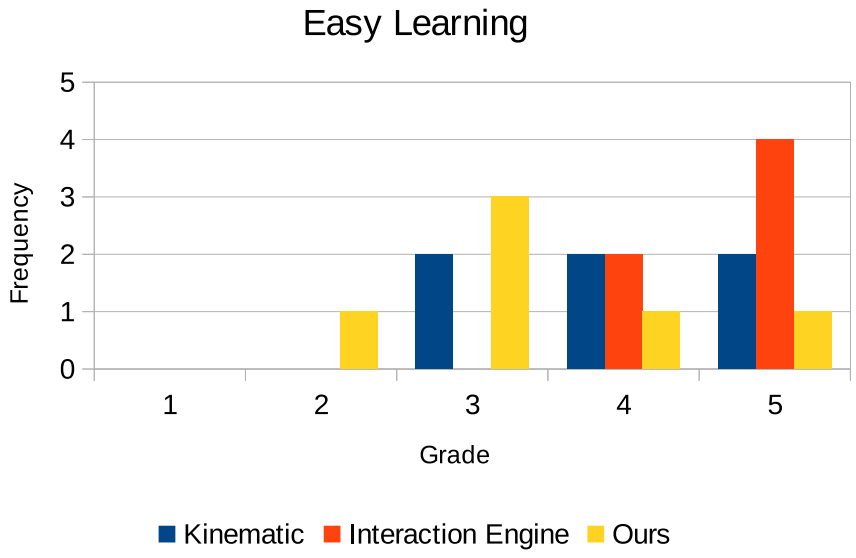


Figure 5.5: Histogram of user answers for the "Easy Learning" question.

5.2 Accuracy

As for the quantitative accuracy evaluation, all 6 users were asked to move a cube from an initial starting position on a chessboard to a specific target location on the chessboard.

The initial position and target position were randomized for each of the 20 trials. We measured the positioning error of the cube with respect to the target location, and the total time required for performing the task. The results of this experiment are shown in Table 5.2. Our approach provides up to 35% higher accuracy compared to the other methods, achieving an average accuracy of less than 10mm. This is due to the more diverse, natural and fine interaction possibilities, such as pushing and pulling, for fine alignment and the realistic consideration of the friction phenomena.

Input method	Positioning error / mm
Our re-implementation of [23]	14.3 ± 8.5 / 12.2
Leap Motion Interaction Engine	11.1 ± 6.9 / 9.6
Our approach	$\mathbf{9.2 \pm 5.5}$ / **8.2**

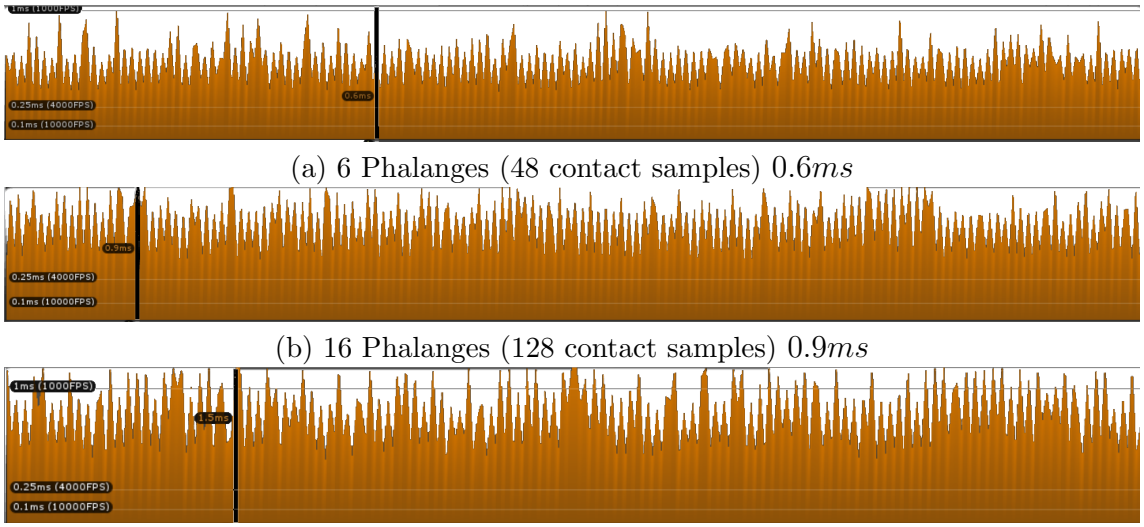
Table 5.2: Quantitative evaluation of interaction accuracy. We denote the average object positioning error with standard deviation and median over 20 trails for 6 different users. Our method achieves the most accurate object positioning, with less than 10mm average error.

Since the users have full control over the exerted forces, objects can be placed with much higher precision. This is a strong indication that our approach can be used for more accurate user interfaces. All methods showed an average interaction time below 10s. While the simpler methods, the Interaction Engine and our re-implementation of [23], showed an average interaction time of 6.6s and 6s respectively, the users had a slightly longer interaction time of 9.9s with our method.

5.3 Performance

We evaluate our method in terms of real-time performance since computational efficiency is one of the most critical aspects for VR/AR integration. The frame rate analysis presented in Fig. 5.6 has been performed with the same hardware setup as reported in Section 4. Each image shows the number of phalanges being in contact with an object, the number of sampled contact points exerting forces (8 samples per contact point), and the computation time over multiple frames.

We state the total computational cost of the physics simulation which includes all physics simulations of the scene, as well as the simulation of our interaction model. The simulation is very fast, taking only $0.6ms$ for 6 phalanges or $1.5ms$ for 32 phalanges. Our performance measurements are significantly faster than the timings reported in [43] ($30ms - 115ms$), who use a more complex friction model (*Coulomb-Contensou*). The evaluation clearly shows the efficiency of our approach which is essential for real-time applications.



(a) 6 Phalanges (48 contact samples) $0.6ms$

(b) 16 Phalanges (128 contact samples) $0.9ms$

(c) 32 Phalanges (256 contact samples) $1.5ms$

Figure 5.6: Performance analysis of the physics computation. The graph illustrates the run-time of the physics simulation with varying number of bones being in contact. Each contact is sampled with 8 contact samples within the patch area. The black bar depicts one frame in the timeline for which we state the simulation time.

5.4 Discussion

The users responded very positively to our physics-based method since there are many interaction possibilities and it feels more realistic and natural. Our experimental tests showed that users preferred to use friction to let the object slide between fingers and the chessboard. They tried to use that interaction also during the tests of the other two methods, but those approaches do not support that, at least not in a similar way.

Users responded less positively to the other two methods because these methods felt very artificial and limited since there is no friction contact possible.

The users rated the naturalness, usefulness and diversity of our method much higher than for the other methods. Overall, 4 out of 6 users preferred our physics-based method over the other two methods.

We conclude that our method is more realistic as it allows for more diverse interactions. Despite the simplicity of the *Coulomb Friction Model*, our approach produces realistic object manipulations and proves to be capable of achieving real-time performance for *VR* and *AR* applications.

6 Conclusions

In this thesis, we presented a novel unconstrained physics-based approach, which produces realistic hand-object interactions in computational expensive environments such as Augmented Reality (AR) and Virtual Reality (VR). We proposed to rely on the *Coulomb Friction Model* to introduce friction and we showed how to efficiently implement it for real-time performance.

Compared to other methods, our approach does not rely on any pre-recorded data. We have aimed to capture the full Degrees of Freedom (DoF) of human hand interactions including dexterous manipulations by finely exerting forces using fingers. Also, our system supports interactions using multiple hands simply by design.

We investigated several recent bare-hand approaches for direct object manipulation to exceed the observed limitations without sacrificing performance. In particular, we successfully solved major problems in the domain of bare-hand interaction with virtual objects. This includes contact point estimation, solving the interpenetration problem, updating surface points with respect to dynamic contact states, stable force exertion, contact force computation, and reasonable visualization. Our method enables convincing physics-based direct object manipulations with real-time performance in the domain of *AR* and *VR*. Through our experiments and evaluation, we showed that it results in a more natural way to interact with virtual objects.

We showed that the *Coulomb Friction Model*, while being a rather simple approximation model for capturing the complexity of real friction phenomena, can still produce convincing and complex object manipulations with proper implementation. Thus, it represents an attractive trade-off in already computational expensive environments.

To develop this approach we have efficiently exploited functionality from State-of-the-Art (SotA) frameworks such as the Unity graphics engine and the integrated PhysX physics engine from Nvidia and built upon them.

The interpenetration problem could be solved with respect to contact point estimation by using the collision detection system of these frameworks. We defined a small threshold to induce early collisions between our hand model and the virtual objects. This threshold has been chosen in such a way that we could receive accurate collisions without any real physical contact between the virtual hand skeleton and virtual objects while not limiting the speed of hand motions in any way. These early collision points allowed us to compute actual contact points directly on the objects' surface without undesired collision resolutions from the engine.

We approached the problem of dynamic surface point updates during contact state with an efficient solution. Our solution exploits the power of the given frameworks with respect to our implementation of the *Coulomb Friction Model* to distinguish between static- and dynamic contacts. By detecting dynamic contacts, we could estimate new contact points directly on the objects surface while considering arbitrary object shape. Thus, an updated surface point may slide realistically along the objects' surface during a slipping contact state but also resides within blocked areas of arbitrary object shapes.

By approximating the property of human skin, which consists of a large amount of contact points, we could achieve stable force control. Instead of using single contact points for exerting the computed contact forces we sampled patches with multiple contacts around the estimated surface points. This allows for stable object interactions such as grasping without perceptible performance decrease.

We showed a way how to compute reasonable contact forces derived from real hand motions and considering normal and tangential components to enable friction along the object surface according to the *Coulomb Friction Model*. Our method showed how exerted contact forces can successfully counteract external forces such as gravity or collision forces.

For the visualization we presented a simple way to render reasonable hand meshes during contact without visual interpenetrations between the hand mesh and virtual objects. We supported intentional interactions by using semi-transparent hand meshes in *VR* to give a better depth perception in contact free state. For *AR* we proposed to use an occlusion mask for a correct depth perception. Additionally, we integrated a simple feedback for *VR* environments by rendering a full opaque hand mesh during contact states and playing a subtle sound effect in *AR* as soon as a contact is established.

Our method uses widely accessible hardware, which increases reproducibility and potential future work. The whole method acts as middleware between the real hand motions derived from the Leap Motion sensor and the actual physics engine.

6.1 Future Work

Despite the promising results, our approach still offers several improvements. One of the very obvious improvements is to integrate a more accurate hand-tracking system. The tracking capabilities of the Leap Motion sensor has significantly improved with the released *Orion* beta update. However, especially for very dexterous manipulations, the accuracy is still rather limited. Since our system is entirely physics-based where the user can freely control the contact forces solely by his real hand motions, tracking mismatches of the hand skeleton can cause undesired interactions. Still, the device has been proven to be a good choice overall due to its wide accessibility and easy integration in modern frameworks, but it should be considered to rely on a more accurate tracking system in future.

Our method produces realistic interactions, though in reality it is easier to control the forces for inducing intended object manipulations. This is because we have a haptic feedback since we literally feel the object, feel its weight, feel its motions and feel the impact of our forces and respond to it immediately. That is why a haptic feedback system [30, 36] would support the force control significantly. There would be other possibilities as well, some sort of guidance system could improve the control over the forces. The guidance mechanic could recognize the intended forces and limit outliers. This may balance the lack of a haptic feedback system.

Extremely thin objects, e.g., a streak of hair or thin paper would need special treatment to be able to interact with them. Our approach relies to some degree on the penetration depth of the virtual bones, thus we need a certain minimum of object volume to determine the magnitude of contact forces.

Another potential improvement is to introduce a more complex and accurate friction model instead of the *Coulomb Friction Model*. We proved that the Coulomb model can produce very convincing object manipulations with proper implementation. This is especially interesting for already computational expensive environments. However, the model is still a simple approximation of real friction phenomena. The degree of realism could be increased by introducing a more complex friction model. Though, the model needs to be selected carefully and must not cause a perceivable decrease in performance. It would be interesting to see how far we could go in terms of the trade-off between performance and realism when integrating more accurate friction models in our system for *AR* or *VR* environments.

6.2 Outlook

We believe that our work will provide a solid basis for the development of unconstrained and natural hand-object interaction in *AR* and especially in *VR* environments, as it acts

as a middleware between the real hand input and commodity graphics engines and builds upon widely accessible hardware.

We are convinced that physics-based simulation is the proper way of how to achieve realistic bare-hand interaction systems for direct object manipulations in a long term. The reason for that is rather simple: introducing an interaction system, which relies on the principles of physics, simulates the process of hand interaction from reality.

For us humans, our hands are the most natural interaction method we can think of. Physics-based methods are the only way, which can provide a general solution by design. As in reality, we are not constrained by how to use our hands on different objects. Every interaction between objects is induced by exerting forces and is only possible due to the phenomena of friction.

What we have to ask ourselves: How could we exploit the full potential of natural hand interaction systems for *VR* and *AR* environments? What is the future of interaction system with respect to *AR* and *VR*? To what degree do we want to change and form the new way of interacting with digital data?

The industry including the Microsoft HoloLens Gaze system [1], HTC Vive controllers [2], and Oculus Rift Touch [3], as well as recent literature [1–3, 8, 12, 17, 20, 21, 23–26, 28, 29, 31, 33, 35–39, 43, 47, 48, 50, 52, 53] indicate a vast increase of interest in new interaction methods and especially hand interaction methods. *VR* is predestinated for games, immersive virtual experiences, simulations and training of all kinds of different professions, e.g., surgery training in the medical sector. We see a strong benefit in this sector for realistic hand-object interactions.

Currently developed *VR* experiences clearly confirm the significance of hand interaction systems. In addition to the realism, physics-based methods represent a general solution for interacting with virtual objects by design. Virtual objects, which are configured with physical properties such as mass, drag, angular drag, static- and dynamic friction coefficients describing the surface material, simply react accordingly on force exertion.

However, one of the most critical aspects is performance as the phenomena of friction is extremely complex in reality as we have discussed in Section 3.4. That is why approximations have to be done to model it for real-time applications. Generally, the whole nature of computer graphics is to imitate realistic behavior using approximations to implement solutions with a fine trade-off between accuracy and performance.

AR applications show also a significant potential for realistic hand-object interaction systems, though the potential might be currently not as strong compared to *VR* considering the bulk of common *AR* applications. In principal, the same aspects are essential

[1] `https://www.microsoft.com/en-us/hololens`
[2] `https://www.vive.com/us/accessory/controller/`
[3] `https://www.oculus.com/rift/`

here. Applications such as games, simulations, training or even modeling have a strong potential in *AR*. It depends strongly on the application itself. More precisely, *AR* environments containing virtual objects, which have realistic properties or any physical relation to reality provide a major potential for natural hand-interaction systems to manipulate objects directly.

These aspects have to be considered for future *AR* and *VR* applications. It is still a long road until we reach the point where we really can claim to have solved virtual hand interaction for direct object manipulation entirely since it is such a complex domain. However, with our approach we could achieve already very promising results while maintaining a stable real-time frame rate.

Due to the vast increase of *AR* and *VR* technology, the interest in hand interaction systems will ascend to an even greater level with forthcoming years. We hope that this thesis drives future development of hand interaction systems for *AR* and *VR* environments since we proposed efficient solutions for major challenges in that domain.

With all that said, realistic hand-object interaction methods will be undoubtedly part of future interaction systems in the realms of *AR* and *VR* and are meant to change the way how we interact with digital data.

A

List of Acronyms

AR	Augmented Reality
CDL	Christian Doppler Laboratory
DCO	Default Contact Offset
DoF	Degrees of Freedom
FoV	Field of View
FPS	Frames per Second
GO	Game Object
HMD	Head Mounted Display
ICD	Inter Camera Distance
ICG	Institute of Computer Graphics and Vision
IPD	Interpupillary Distance
LoD	Level of Detail
MR	Mixed Reality
OSTMHD	Optical See Through Head Mounted Display
SL	Structured Light
SLAM	Simultaneous Localization and Mapping
SotA	State-of-the-Art
VCD	Virtual Camera Distance
VR	Virtual Reality

Bibliography

[1] Avila, L. and Bailey, M. (2016). Augment your reality. *CGA*, 36(1):6–7. (page 5, 70)

[2] Benko, H., Jota, R., and Wilson, A. (2012). Miragetable: freehand interaction on a projected augmented reality tabletop. In *CHI*. (page 1)

[3] Borst, C. W. and Indugula, A. P. (2005). Realistic virtual grasping. In *VR*. (page 2, 5, 8, 70)

[4] Borst, C. W. and Indugula, A. P. (2006). A spring model for whole-hand virtual grasping. *Presence: Teleoperators and Virtual Environments*, 15(1):47–61. (page 24)

[5] Bracegirdle, A., Mitrovic, T., and Mathews, M. (2014). Investigating the usability of the leap motion controller: Gesture-based interaction with a 3d virtual environment. Technical report, University of Canterbury, NZ. (page 5)

[6] Buchmann, V., Violich, S., Billinghurst, M., and Cockburn, A. (2004). Fingartips: gesture based direct manipulation in augmented reality. In *Proceedings of International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*. (page 2, 5)

[7] Chen, W.-H., Ballance, D. J., Gawthrop, P. J., and O'Reilly, J. (2000). A nonlinear disturbance observer for robotic manipulators. *IEEE Transactions on industrial Electronics*, 47(4):932–938. (page 17, 18)

[8] Egger, J., Gall, M., Wallner, J., Boechat, P., Hann, A., Li, X., Chen, X., and Schmalstieg, D. (2017). Integration of the htc vive into the medical platform mevislab. In *SPIE Medical Imaging*. (page 1, 70)

[9] Erez, T., Tassa, Y., and Todorov, E. (2015). Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *Proceedings of the International Conference for Robotics and Automation*. (page 7, 36)

[10] Gafvert, M. (1997). Comparisons of two dynamic friction models. In *Control Applications, 1997., Proceedings of the 1997 IEEE International Conference on*, pages 386–391. IEEE. (page 17)

[11] Gao, J., Luedtke, W., Gourdon, D., Ruths, M., Israelachvili, J., and Landman, U. (2004). Frictional forces and amontons' law: from the molecular to the macroscopic scale. (page 17)

[12] Garre, C., Hernandez, F., Gracia, A., and Otaduy, M. A. (2011). Interactive simulation of a deformable hand for haptic rendering. In *Proc. of World Haptics Conference*. IEEE. (page 8, 70)

[13] Gordon, C. C., Churchill, T., Clauser, C. E., Bradtmiller, B., and McConville, J. T. (1989). Anthropometric survey of us army personnel: methods and summary statistics 1988. Technical report, Anthropology Research Project Inc Yellow Springs OH. (page 42)

[14] Gourret, J.-P., Thalmann, N. M., and Thalmann, D. (1989). Simulation of object and human skin formations in a grasping task. *SIGGRAPH*, 23(3):21–30. (page 22)

[15] Guna, J., Jakus, G., Pogačnik, M., Tomažič, S., and Sodnik, J. (2014). An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors*, 14(2):3702–3720. (page 1, 5)

[16] Hand, C. (1997). A survey of 3d interaction techniques. *CGF*, 16(5):269–281. (page 5)

[17] Hilliges, O., Kim, D., Izadi, S., Weiss, M., and Wilson, A. (2012). Holodesk: direct 3d interactions with a situated see-through display. In *CHI*. (page 1, 2, 5, 7, 70)

[18] Höll, M., Heran, N., and Lepetit, V. (2016). Augmented reality oculus rift. *arXiv preprint arXiv:1604.08848*. (page 13)

[19] Höll, M. and Lepetit, V. (2017). Monocular lsd-slam integration within ar system. *arXiv preprint arXiv:1702.02514*. (page 42)

[20] Jacobs, J. and Froehlich, B. (2011). A soft hand model for physically-based manipulation of virtual objects. In *VR*. (page 2, 5, 7, 70)

[21] Jacobs, J., Stengel, M., and Froehlich, B. (2012). A generalized god-object method for plausible finger-based interactions in virtual environments. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 43–51. IEEE. (page 21, 70)

[22] Kelly, R., Llamas, J., and Campa, R. (2000). A measurement procedure for viscous and coulomb friction. *IEEE Transactions on instrumentation and measurement*, 49(4):857–861. (page 17)

[23] Kim, J. and Park, J. (2015). Physics-based hand interaction with virtual objects. In *ICRA*. (page 5, 9, 45, 53, 59, 61, 64, 70)

[24] Kim, J.-S. and Park, J.-M. (2016). Direct and realistic handover of a virtual object. In *IROS*. (page 5, 9, 47, 53)

[25] Li, Y., Fu, J. L., and Pollard, N. S. (2007). Data-driven grasp synthesis using shape matching and task-based pruning. *TVCG*, 13(4):732–747. (page 6, 7)

[26] Marin, G., Dominio, F., and Zanuttigh, P. (2014). Hand gesture recognition with leap motion and kinect devices. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 1565–1569. IEEE. (page 70)

[27] Miller, A. T., Knoop, S., Christensen, H. I., and Allen, P. K. (2003). Automatic grasp planning using shape primitives. In *ICRA*. (page 6, 7)

[28] Moehring, M. and Froehlich, B. (2011). Effective manipulation of virtual objects within arm's reach. In *VR*. (page 2, 5, 70)

[29] Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., and Kautz, J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *CVPR*. (page 6, 70)

[30] Murayama, J., Bougrila, L., Luo, Y., Akahane, K., Hasegawa, S., Hirsbrunner, B., and Sato, M. (2004). Spidar g&g: a two-handed haptic interface for bimanual vr interaction. In *Proceedings of EuroHaptics*, volume 2004, pages 138–146. Citeseer. (page 69)

[31] Nasim, K. and Kim, Y. J. (2016). Physics-based interactive virtual grasping. In *Proc. of HCI Korea*. (page 2, 5, 8, 11, 32, 70)

[32] Oberweger, M., Wohlhart, P., and Lepetit, V. (2015). Hands deep in Deep Learning for hand pose estimation. In *Proc. of Computer Vision Winter Workshop*. (page 1)

[33] Ortega, M., Redon, S., and Coquillart, S. (2007). A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties. *IEEE transactions on visualization and computer graphics*, 13(3):458–469. (page 21, 70)

[34] Pavlovic, V. I., Sharma, R., and Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: A review. *Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):677–695. (page 5)

[35] Perez, A. G., Cirio, G., Hernandez, F., Garre, C., and Otaduy, M. A. (2013). Strain limiting for soft finger contact simulation. In *Proc. of World Haptics Conference*. IEEE. (page 8, 70)

[36] Pfeiffer, M., Schneegass, S., Alt, F., and Rohs, M. (2014). Let me grab this: A comparison of ems and vibration for haptic feedback in free-hand interaction. In *Proceedings of the 5th augmented human international conference*, page 48. ACM. (page 69)

[37] Pollard, N. S. and Zordan, V. B. (2005). Physically based grasping control from example. In *SCA*. (page 5, 9)

[38] Prachyabrued, M. and Borst, C. W. (2012). Virtual grasp release method and evaluation. *International Journal of Human-Computer Studies*, 70(11):828–848. (page 2, 6)

[39] Rautaray, S. S. and Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction: a survey. *JAIR*, 43(1):1–54. (page 6, 70)

[40] Rijpkema, H. and Girard, M. (1991). Computer animation of knowledge-based human grasping. In *SIGGRAPH*. (page 2, 6, 7)

[41] Stribeck, R. and Schröter, M. (1903). *Die wesentlichen Eigenschaften der Gleit-und Rollenlager: Untersuchung einer Tandem-Verbundmaschine von 1000 PS*. Springer. (page 18)

[42] Sturman, D. J., Zeltzer, D., and Pieper, S. (1989). Hands-on interaction with virtual environments. In *UIST*. (page 5)

[43] Talvas, A., Marchal, M., Duriez, C., and Otaduy, M. A. (2015). Aggregate constraints for virtual manipulation with soft fingers. *IEEE transactions on visualization and computer graphics*, 21(4):452–461. (page 8, 64, 70)

[44] Taylor, J., Bordeaux, L., Cashman, T., Corish, B., Keskin, C., Sharp, T., Soto, E., Sweeney, D., Valentin, J., Luff, B., et al. (2016). Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *TOG*, 35(4):143. (page 7)

[45] Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *IROS*. (page 3, 36)

[46] Von Hardenberg, C. and Bérard, F. (2001). Bare-hand human-computer interaction. In *Perceptive User Interfaces*. (page 1, 5)

[47] Wingrave, C. A., Williamson, B., Varcholik, P. D., Rose, J., Miller, A., Charbonneau, E., Bott, J., and LaViola Jr, J. J. (2010). The wiimote and beyond: Spatially convenient devices for 3d user interfaces. *CGA*, 30(2):71–85. (page 1, 70)

[48] Yim, D., Loison, G. N., Fard, F. H., Chan, E., McAllister, A., and Maurer, F. (2016). Gesture-driven interactions on a virtual hologram in mixed reality. In *Proceedings of ACM Companion on Interactive Surfaces and Spaces*. (page 2, 5, 70)

[49] Young, D. P., Melvin, R. G., Bieterman, M. B., Johnson, F. T., Samant, S. S., and Bussoletti, J. E. (1991). A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 92(1):1–66. (page 36)

[50] Zell, E., Aliaga, C., Jarabo, A., Zibrek, K., Gutierrez, D., McDonnell, R., and Botsch, M. (2015). To stylize or not to stylize?: The effect of shape and material stylization on the perception of computer-generated faces. *TOG*, 34(6):184:1–184:12. (page 32, 70)

[51] Zhang, M. and Mak, A. (1999). In vivo friction properties of human skin. *Prosthetics and orthotics International*, 23(2):135–141. (page 26)

[52] Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE Multimedia*, 19(2):4–10. (page 5, 70)

[53] Zhao, W., Zhang, J., Min, J., and Chai, J. (2013). Robust realtime physics-based motion control for human grasping. *TOG*, 32(6):207. (page 5, 9, 10, 70)

[54] Zimmermann, C. and Brox, T. (2017). Learning to estimate 3D and pose from single RGB images. In *ICCV*. (page 1)