Investigating Constrained Objects in AR for Validation of Real-life Models

Brandon Vassion

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2023

Committee:

Kelvin Sung

Yusuf Pisan

Michael Stiber

Program Authorized to Offer Degree:

Computer Science and Software Engineering

University of Washington

Abstract

Investigating Constrained Objects in AR for Validation of Real-life Models

Brandon Vassion

Chair of the Supervisory Committee:

Kelvin Sung

Department of Computer Science and Software Engineering

Augmented Reality (AR) studies the approaches that enhance reality by integrating virtual content into the physical environment in real-time. In the simplest form, virtual objects in the physical environment are stationary, where AR applications serve as powerful tools for visualization. The support of interaction with objects in the environment brings the AR application from being passive for observing the augmented world to one where the user can actively explore. When the interactions follow intuitive physical world constraints, an AR application, or constraint-based AR, can immerse users in a realistic augmented world.

We categorize existing constraint-based AR by the relationship between and interaction of the objects being constrained: virtual objects constrained by virtual objects, physical by virtual, and virtual by physical. This straightforward classification provides insights into the types of and potentials for useful applications. For example, virtual by virtual can describe the pages of a virtual book being constrained where the corresponding interaction would be the

flipping of the virtual pages. In contrast, physical by virtual would mean placing a physical coffee cup over the virtual book. Lastly, virtual by physical would be placing and pushing the virtual book on an actual physical desktop. The subtle and yet crucial differences are that in the first case, the objects and the interactions can also be carried out in a pure virtual 3D world, physical by virtual has practical implementation challenges, and that, virtual by physical presents an interesting opportunity for immersing and engaging users.

This thesis investigates using virtual by physical constraint-based AR to validate the functionality and visuals of real-life models. We observe and identify common and representative real-world interaction constraints to include: 1D sliding, 2D planar sliding, hinged rotation, and the potential for combining these constraints. The thesis examines the functionality, interactability, and integration of these constraints in practical applications, in this case, a home decoration setting. With the results from an initial technology investigation, aiming to achieve accuracy and reliability in interactions, we have chosen marker-based AR through Vuforia with Unity3D. We have derived a systematic workflow for creation and have demonstrated successful integration of virtual objects into the real world with relevant constraints by corresponding physical objects. Our prototype results are various versions of an augmented room with distinct decorative virtual objects that are constrained by relevant physical objects where the interactions are intuitive and integrations essentially seamless. These rooms support multiple constrained objects functioning in the same environment.

Our new contributions to our field are as follows: Our categorization points to a well-defined AR application domain, virtual by physical, for investigation, where the success of the augmented rooms demonstrates the effectiveness of this category of constraint-based AR applications in validating functionality and visuals. Our formulated workflow for constructing

virtual by physical constraint-based AR applications serves as an efficient and effective template for future investigations into this domain. Finally, we expand upon previous results of working with constraints in AR by demonstrating the feasibility of combining multiple types of constraints into a new constraint type.

Table Of Contents

Chapter 1 Introduction

Validating the visuals and functionality of an object is a process most perform when they acquire something new. It might be as simple as trying on new clothes, or it can be more involved such as checking to make sure a newly installed cabinet works. Regardless of the object being validated, there is a requirement for the physical object to be present. In many cases this can be cumbersome when: there is a need to validate multiple objects, it is challenging to build the object, or when it involves high costs. There is then a need to examine and interact with these objects without physically having them. To accomplish this we can instead use virtual versions of these objects to enhance real-world objects for validation.

Augmented Reality (AR) is a field of study that examines approaches to enhance our reality by overlaying virtual content onto our physical environment in real-time. When a user views the physical world through the AR devices, virtual objects augment how they see the environment. There are a variety of application fields such as gaming [1], education [2], healthcare [3], and retail [4]. By presenting the virtual objects in the physical environment, AR can be a powerful tool for validation, as it is capable of the two requirements needed for validation, the ability to visually examine, and the ability to physically interact.

One of AR's strongest aspects is in its powerful ability to visualize. For example, in construction [5], AR can be used to overlay the insides of the ground to prevent accidental drilling into pipes or wires as we can see in Figure 1.1. Another example is in healthcare, AR can be used to visualize the anatomy of the human body to allow medical professionals to see things such as internal organs in 3D [6]. In the case of validation, this ability can be used to examine any virtually-enhanced real-world object to confirm that its visuals are correct.

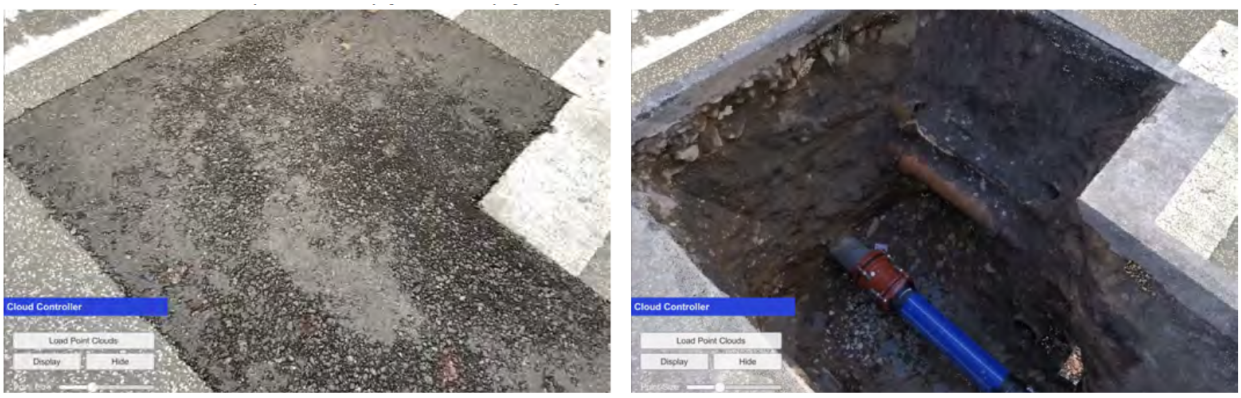

Figure 1.1: The real world with no Augmented Reality(left). Same location as seen with Augmented information overlaid to see into the ground [5].

There are many different ways in which users can interact with an enhanced world in AR, such as using natural gestures [7], voice commands [8], direct interaction [9], or screen controls [10]. For example, Figure 1.2 shows using direct interaction to assemble a virtual puzzle.

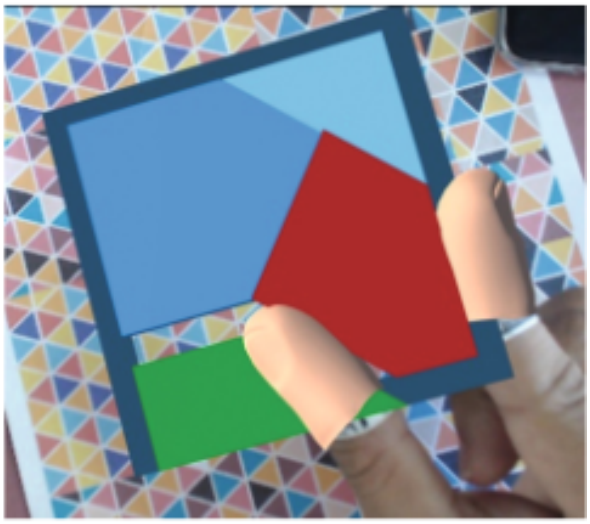

Figure 1.2: Virtual puzzle being put together by hand in AR [9]

Supporting interactions with virtual objects that follow intuitive real-world constraints can further expand on AR's ability to support validation through interaction. For example, taking the chest in Figure 1.3, we could have it sitting on a table and we could interact by pushing it, picking it up, moving it around, and rotating it. However, with the addition of constraints we can take that same chest and add a hinge constraint to the lid, allowing for an intuitive interaction of opening and closing the chest. In this way, support for constrained motion allows for the delivery of more complex interactions that can be used in validating objects

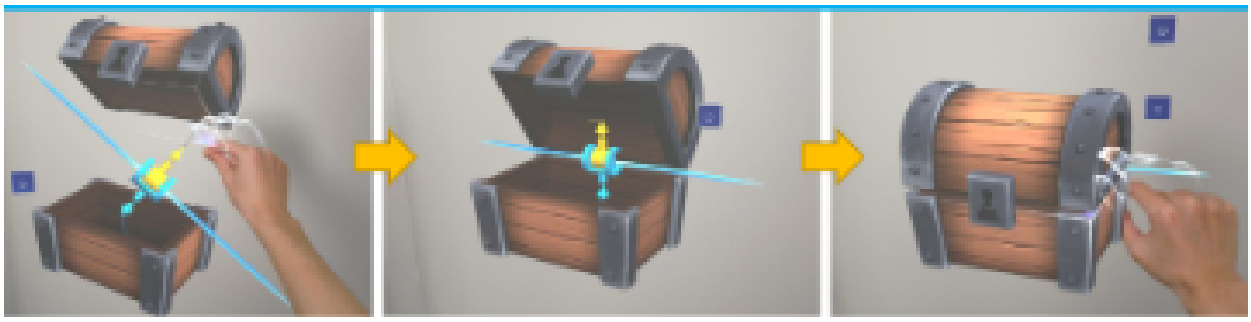

Figure 1.3: Hinge constraint being added between lid and base to create a functioning chest [7]

It is clear that AR-based systems can be effective in assisting our investigation into validation of the visuals and functionality of virtually-enhanced real-world objects. However, the immense size of this problem domain and solution space necessitate a categorization of the existing results. Based on the enhancement mechanism or constraints, and the relationships between the

virtual and real-world objects, we propose a straightforward and yet effective solution framework to categorize these results. The categorization shed insights and identified an important and well-defined sub-domain, virtual-enhancements constrained by real-world objects, for investigation.

To study the effectiveness of virtual objects being constrained by real world objects in supporting visual and functional validation, we chose to use three main types of constraints as introduced by Wang et al. [7]: the sliding constraint, the planar constraint, and the hinge constraint, as well as a more complex constraint composed of multiple constraints. We chose these constraints because they represent typical constraints in life such as: an object on rails (sliding), objects on a surface (planar), or an object rotating about an axis (hinge). To help examine these constraints we needed an environment in which to do so. We chose to use a home decoration setting because it allowed a straightforward way to integrate these constraints.

To accomplish the investigation, we began studying the interactions with the constraints in a virtual world independently from the physical world to understand how to implement them properly when introduced into AR. We also prototyped limited versions of each constraint in AR to learn how to work and interact with virtually generated objects in the physical world. Examining the use of multiple constraints in the same virtual environment allowed us to then integrate it into the AR environment.

In our work we wanted to investigate validating the visuals and functionality of virtually-enhanced real-world models. Our investigation led to three main contributions. First, it led to our new categorization of constraint-based AR, which allowed us to identify a well-defined problem within one of the sub-domains. Second, we proposed an effective step-by-step process for setting up physically-constrained virtual objects in the real world. Third, we were able to demonstrate the support of multiple constraints as well as the ability to combine the use of different constraints on a single object.

Chapter 2 of this report covers the related works pertaining to the various forms of constraint-based AR. Chapter 3 describes the technology and initial investigations. Chapter 4 describes our implementation of the constraint-based AR objects. Chapter 5 discusses the results of our implementation. Chapter 6 discusses the work along with limitations and possible future works.

Chapter 2 Related Works

In this chapter we examine existing AR applications using our categorization based on the relationship between and interaction of the objects being constrained: virtual constrained by virtual, physical constrained by virtual, and virtual constrained by physical.

2.1 Virtual constrained by Virtual

In one common form of augmented reality, the generated virtual objects are only constrained by other virtual objects and are unaffected by any physical objects in the surroundings. Zugara's [11] Interactive Virtual Object (IVO) Engine is an example where they have created a demo of a virtual solar system. In this demo the planets were constrained with respect to the position of the sun; as the sun was moved around, all the other planets moved in turn. In this case, the virtual objects have a frame of reference with the AR device and the user, but none with the surrounding space. Wang et al. [12] worked on using augmented reality for assembly of parts by hand, as shown in Figure 2.1. In their work, as virtual objects are being put together they become constrained to one another and become one whole object. While largely unaware of the physical space, these virtual objects have some reference of the surroundings, aside from just floating in space, they can be placed on a surface during the assembly process. Boonbrahm and Kaewrat [13] similarly used AR for assembly by hand, instead using a fully marker-based approach. Their approach involved attaching markers to their fingers to interact with the constrained objects.

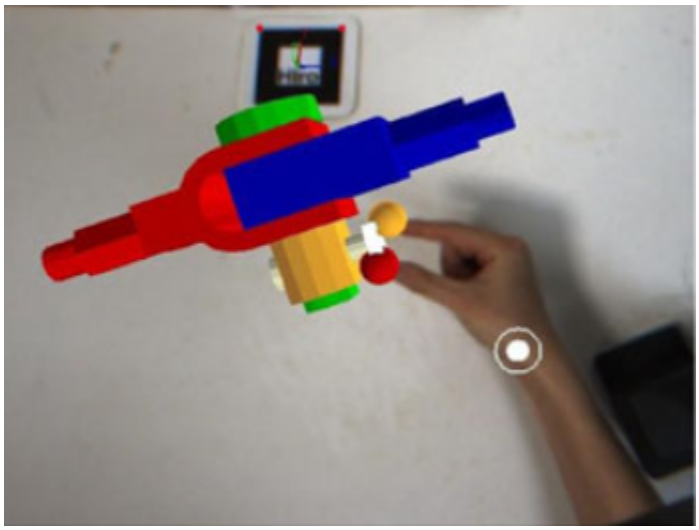

Figure 2.1 - Free floating virtual object being assembled in AR [12]

2.2 Physical constrained by Virtual

In another category of constraint-based AR, the virtual world imposes constraints onto objects in the physical world. Not much work has been done in this area, but Patten and Ishii [14] developed a tabletop tangible interface using an array of electromagnets to accomplish this. Figure 2.2 shows the virtual constraining the physical, where three physical objects are constrained by virtual objects to be at an equal distance from one another. Once one of the objects moves, the others will be moved to maintain the virtual constraint as if they were physically connected.

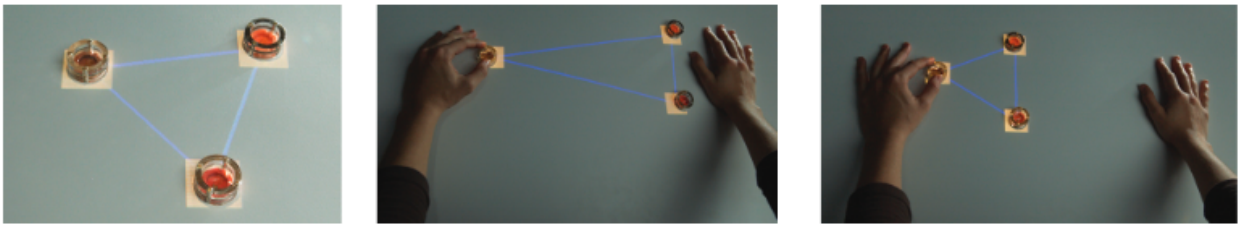

Figure 2.2 - Physical object dragged outside of constraint; connected objects respond to obey constraint [14]

2.3 Virtual constrained by Physical

The last category of constraint-based AR is where the physical world constrains the virtual object. This type of constraint-based AR can be broken down into two categories: non-interactable and interactable.

2.3.1 Non-interactable

In the non-interactable category, although there is still an interaction with the physical world via the constraint placed on the virtual object, the virtual object is not interactable by the user once the constraint is in place. This kind of AR is commonly used for visualization, for example in AR furniture apps such as IKEA Place [4] allow the user to preview what a piece of furniture would look like in their homes. Once the furniture is placed, it becomes constrained to the floor and no longer moves relative to the AR device; to move the object once more, the constraint must be removed. An et al. [15] developed ARShoe, in which virtual shoes are constrained to overlay the user's foot, allowing them to see what it would look like wearing them without having to go to a store. Figure 2.3 shows the results of ARshoe overlaying the same shoe on four different people. In a more involved use of the idea of physical on virtual constraints, Nuernberger et al. [16] developed a prototype that involved a high amount of awareness of the physical world. In their work they were able to detect real world edges and planar surfaces, and from there were able to constrain virtual objects to them.

Figure 2.3 - Virtual shoes overlaid onto user's feet [15]

2.3.2 Interactable

Opposite of the non-interactable is AR in which the physical world constrains the virtual and is also interactable by the user. Wang et al. [7] took this idea and created an authoring system for freehand gesture AR. Their work introduced additional mechanical constraints for the virtual objects, such as hinge and sliding joints, which allowed for more types of interaction between virtual and physical objects. This work was mainly focused on the techniques itself; others, such as Moehring and Froehlich [17], took it a step further and began applying it for functional validation. Their work tested functional constraints of interior parts of a car such as the rear-view mirror, using virtual versions. Using the rear-view mirror as an example, they gave it a constraint of a ball and socket joint and interacted with the mirror to confirm the functionality of the constraint. Figure 2.4 shows the motions that the mirror was capable of under the constraint.

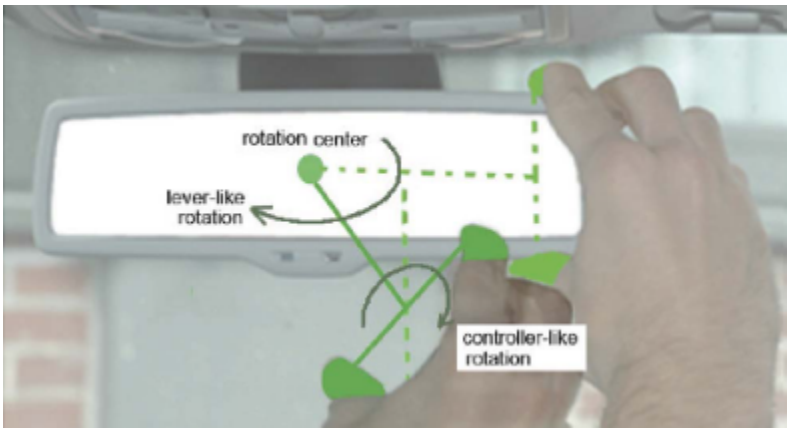

Figure 2.4 - Different interactions on rear view mirror under ball and socket joint constraint [17]

In summary, this literature review reveals the different aspects of these categories of constraint-based AR. In the virtual by virtual case, although it can be intractable by the user, it lacks almost any interaction with the physical world. Virtual by physical contains both

interaction with the user and the environment but requires cumbersome setups to facilitate the constraints. Further, in the non-interactable physical by virtual, the lack of interaction limits the type of functionality that can be worked with. Conversely, interactable physical by virtual is both interactable by the user and has interaction with the physical environment. We believe that the ability to interact with complex constraints can allow for stronger visual and functional validation.

Chapter 3 Technology and Investigation

In this chapter we first discuss the basic requirements for the thesis as well as the tools and approaches used to satisfy the requirements. Then we examine the technology investigation performed, along with the key lessons learned during the investigation. Finally, we present the criteria by which we determine the success of our work.

The purpose of this thesis is to investigate the use of virtual by physical constraint-based AR as a means for visual and functional validation. From this we begin by defining the requirements needed to accomplish our goal.

Basic Requirements:

- Framework or engine for handling AR with multiple objects
- Methods to create virtual objects and constraints
- An AR capable device
- Supports for visually and functionally believable interactions

Additionally, to work with constraint-based AR requires an underlying understanding of AR itself. As we had no experience or knowledge of working with AR prior to this thesis, we needed to perform an investigation into the technology so that we could properly work with constraint-based AR. The investigation also allows us to derive a relevant set of requirements for the thesis.

3.1 Tools and Approaches

The first problem we needed to consider was the method of AR to use, marker-based or markerless AR [18]. We chose the marker-based approach. This approach uses special physical markers that the AR device's camera can detect to generate the relevant AR objects in the physical environment. The advantage of this approach is in the stability when tracking and high accuracy of the markers. In contrast, the markerless approach scans the physical environment to determine where to place AR content. Although the markless approach provides a high level of flexibility this is less desired for our work.

Another decision was how to support a Marker-based approach. In this case, we chose the Vuforia AR engine [19]. Boonbrahm and Kaewrat [13] influenced this decision as their work built with Vuforia involved a method for interaction similar to the one we planned to use. In addition, it is an engine where its marker-based AR provides many useful tools, such as different types of markers and multiple marker tracking support. Options for AR platforms such as ARCore and ARKit are available for their respective mobile platforms, Android and iOS. However, Vuforia leverages both platforms when run on compatible devices, allowing for the extensive marker-based library that Vuforia provides while benefiting from the features of the other two platforms.

To handle the 3D objects that would be generated in AR, we chose to use Unity3D. Unity3D is a game engine for developing 2D and 3D video game applications. Unity has three advantages: 1) Vuforia provides full integration with Unity to make developing 3D AR applications easier. 2) Our familiarity with Unity allowed us to spend less time learning the 3D system and more time understanding the new AR system. 3) The physics engine component of Unity allows for a straightforward implementation of our desired constraints.

The next requirement is an AR capable device. Because we wanted a device that the general public could use, we decided on a smartphone. We chose the Galaxy S22, an Android device. We chose to go with an Android device because the alternative of an iOS device presented barriers to development which the Android device did not have.

In order to satisfy our last requirement of visually and functionally believable interactions, we decided on a direct interaction approach. There are two ways to handle interaction in AR: direct and indirect. In a direct approach the user interacts with virtual objects by using parts of their body (e.g. hand) or by using another physical object. Conversely, indirect approaches involve using screen controls, or voice commands to interact. These however take the interaction away from the space of the objects, where the direct approach performs the interaction in the same space as the virtual objects. Furthermore, users in the physical world interact with objects by actions such as grabbing and pushing; because of this, we wanted to replicate these to create an intuitive interaction.

3.2 Technology Investigation

For this thesis an understanding of working with and interacting with AR was a necessity. To learn the skills needed, we performed an investigation on the technology involved.

Investigation goals:

- Understand basics of AR
- Learn how markers are tracked
- Examine the details of supporting direct interaction
- Discover limitations of the System

First, the investigation began with working with the basic features of Vuforia. In Figure 3.1, the left image shows the created physical image and cylindrical markers, and the right image shows placement of simple virtual objects on these markers.

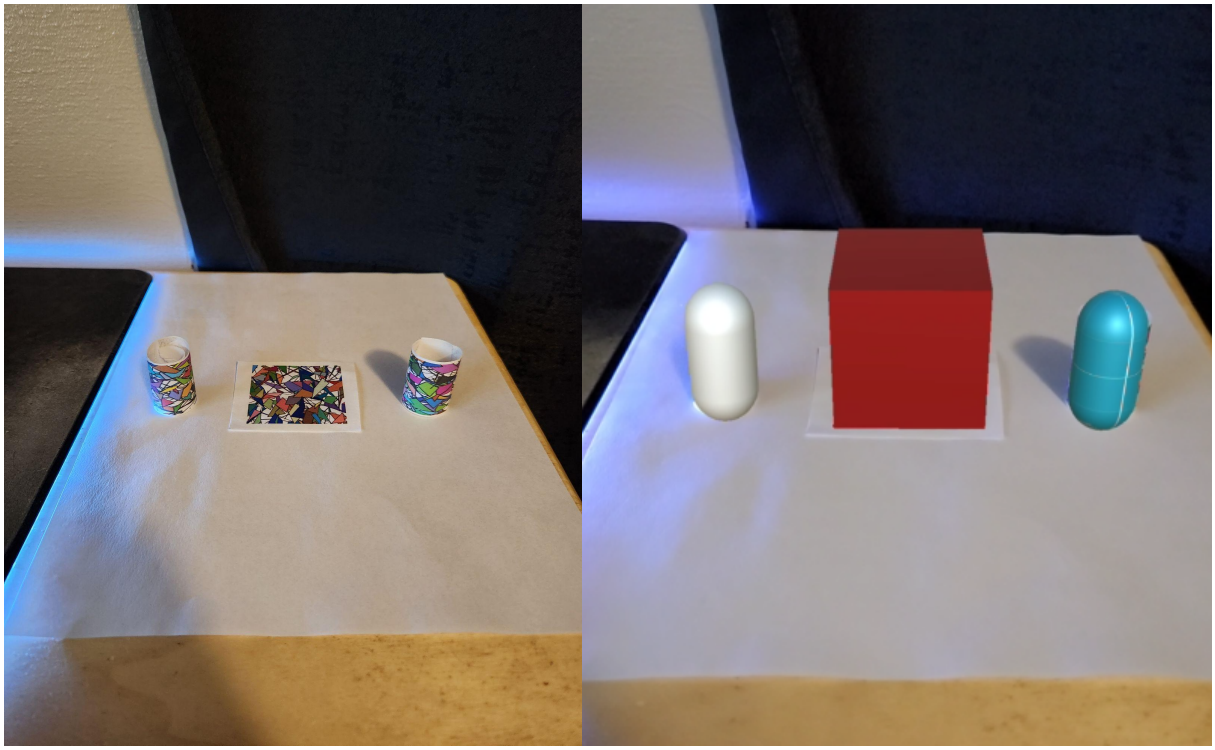

Figure 3.1: Image and cylinder targets (left) and the virtual objects generated by them(right)

Second, we focused the investigation on the different types of modes Vuforia had for tracking markers. The default tracking mode is the basic track mode, where the virtual objects are only displayed if the camera is actively identifying the marker. We found this mode to be most useful for the markers attached to the fingers. Another mode is the extended tracking mode. This mode not only displays the objects when the marker is in sight, but also in certain situations when it is out of view or covered up. We found this mode to be useful when we needed multiple objects to be visible while moving around constantly. Figure 3.2 shows an example of both tracking modes when a marker is covered while displaying the same object.

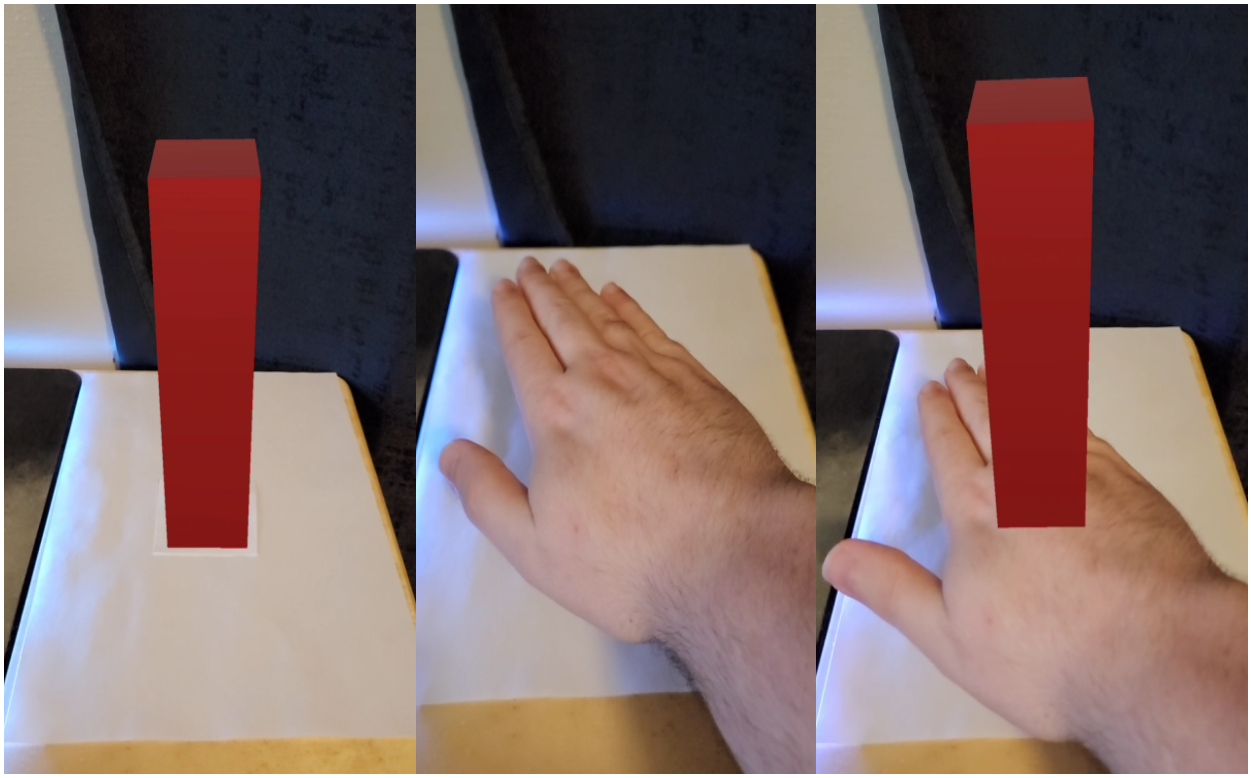

Figure 3.2: The difference between the basic tracking(middle) and the extended tracking(right)

Third, we needed to determine how to handle the direct interaction. Our initial plan began with attaching a cylindrical marker to both the thumb and index finger, and then generating virtual fingers with which to touch the objects with. The approach began with the ability to push, grab, and pick up objects with two fingers as seen in Figure 3.3. When working with simple objects we found this approach to work well, but when working with the more complex constrained objects, the two-finger approach became a more cumbersome process. When dealing with especially large or small objects it was difficult to interact with two fingers, and instead the one finger approach presented itself to be a more usable approach. Since the one finger approach worked well in both normal operations and in edge cases, we decided to adopt the one finger approach for our work.

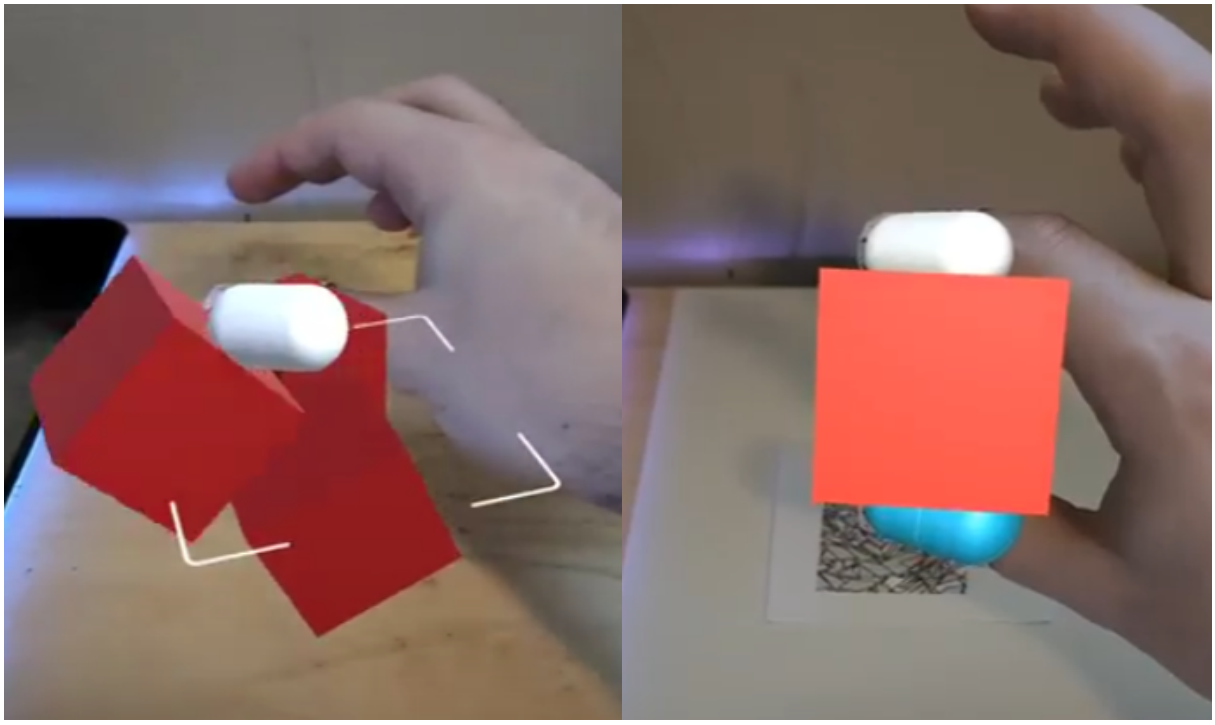

Figure 3.3: Interacting with the virtual cubes by pushing and picking them up

Lastly, three limitations were identified through this initial investigation. The first limitation is on the ability of Vuforia tracking movements. When moving the marker on the finger while holding the camera steady, there is a high chance of the device failing to detect the marker when the marker is moving faster than approximately 0.2 m/s. To avoid dropping detection we observed that moving the camera in sync with the marker allows moving up to approximately 0.9 m/s. The average speed at which a human moves their hands has been measured to be around 0.6 m/s, therefore in most scenarios, moving in sync allows for quicker movement [20].

The second limitation is on the relationship between objects being tracked and the background behind the object. With highly cluttered backgrounds, loss of tracking occurs frequently, due to Vuforia being unable to properly identify the marker. Empty backgrounds can cause Vuforia to be unable to determine the proper orientation of the generated virtual objects, which leads to the object being improperly displayed. In order to avoid these issues, locations with balanced backgrounds are needed.

The final limitation is occlusion of physical objects by the virtual ones. When generating virtual objects Vuforia always presents them in the forefront regardless of whether a physical object is in front or behind it. Occlusion in AR is a large field of study [21] and is not a focus in our work. Therefore, rather than attempting to fix the issues with occlusion, we instead focus on reducing situations where it occurs as much as possible.

3.2.1 Key Lessons

Below are key lessons that were learned through the investigation process:

- Marker images must be of high quality through distinct features for improved detection by AR devices.
- The basic tracking mode should be used with markers that move; the extended tracking mode should be used with markers that are static.
- Occlusion of virtual objects over physical objects should be avoided by keeping virtual objects in front.
- Implementations that function correctly in the editor must be explicitly verified in the actual AR application.
- AR scene setup is an iterative process as detailed in Chapter 4.3.

3.3 Evaluation Criteria

We want to build interactive AR scenes using virtual objects constrained to physical objects for visual and functional validation purposes. Below we define the criteria used to evaluate the success of accomplishing this goal.

Constraints Technical Criteria

- Completeness: Functionality, interaction, and limits of constraints operate intuitively
- Integration: Constrained virtual objects visually and functionally blend with surrounding physical environment
- Scalability: Support multiple constraints in the same environment and allow for combination of constraints.

Virtual by Physical Categorization Criterion

- Verification: Demonstrate Virtual by Physical categorization is meaningful through practical applications in the real world

Chapter 4 Implementation

In this chapter we cover the implementation details of this thesis. Section 4.1 will cover the implementation of the constraints. Section 4.2 will cover the details of interaction. Section 4.3 will cover the constrained AR experience process.

4.1 Implementation of constraints

For this thesis we worked with four different types of constraints. In this section we will discuss how each constraint was implemented.

4.1.1 Slide Constraint

The slide constraint is a constraint that restricts the movement of the constrained object to a single linear axis and restricts any rotation. We create this constraint by using Unity's Configurable Joint script. Once the script is attached to the object that we want to put under the constraint, there are three key parameters that we need to use to facilitate the slide constraint behavior:

- Connected Body - Determines which object the object will be constrained to.
- Degree of Freedom controls - Determines the restrictions on the three translational directions and the three rotational directions.
    - Free - Unrestricted motion and rotation along axis
    - Limited - Motion allowed within set limits
    - Locked - No motion or rotation allowed
- Linear Limit Properties - Sets the limits for all limited linear directions

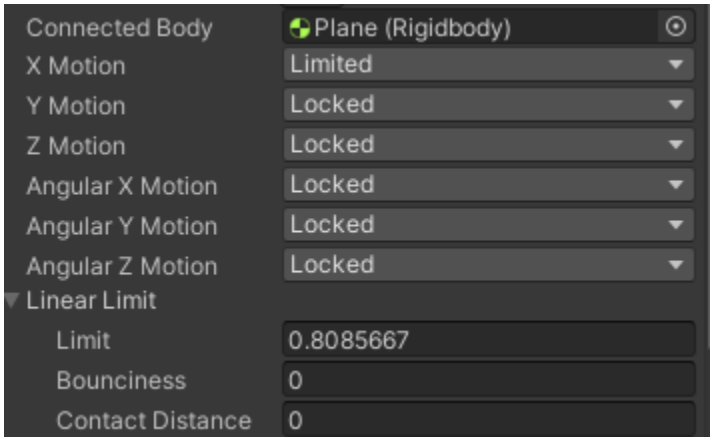

Figure 4.1: Configuration of parameters in Unity editor for a slide constraint

When configuring the constraint, the Connected Body is either the marker object or child of the marker object that is static, in Figure 4.1 this is set to a plane that is a child of a marker object. For the degrees of freedom, one of the translational directions is either set to be Free or Limited, while the other translational directions and all angular directions are set to Locked. In cases

where the translational direction is set to Limited, we also set the limit on how far we allow the object to move, as well as the bounciness and contact distance to the edges. Figure 4.2 shows an example of a slide constraint with a linear limit on a cube.

Figure 4.2: Cube with Slide constraint that has a linear limit with no bounciness

4.1.2 Planar Constraint

The planar constraint is a constraint which restricts the movement of the constrained object to only be able to move within a two-dimensional plane while also preventing any rotation. Similar to the Slide constraint, our work accomplishes this constraint by using Unity's Configurable Joint script. The planar constraint also shares the same three key parameters as the slide constraint, but there are usage and configuration differences.

Figure 4.3: Configuration of parameters in Unity editor for a circular planar constraint

For the planar constraint we have two possible ways to set up this constraint depending on how we choose to limit the motion within the 2D plane. We have the option to either restrict the motion to be constrained in a circular area or in a rectangular area. In the case of a circular area we use a single Configurable Joint with two translational directions set to Limited as displayed in

Figure 4.3. In this scenario the limit defines the radius of the circle that the object is constrained to. An example of this circular planar constraint is shown in Figure 4.4.

Figure 4.4: Cube with Planar constraint that is limited in a circular area

In the case of the rectangular area, we instead use two Configurable Joint scripts. Figure 4.5 shows a configuration of the scripts for this constraint. Each script is responsible for constraining a single direction's motion while leaving the other direction free for the other script to constrain. Since each script is only responsible for a single direction, the applied limits can be different, resulting in the bounds defining the rectangular area.

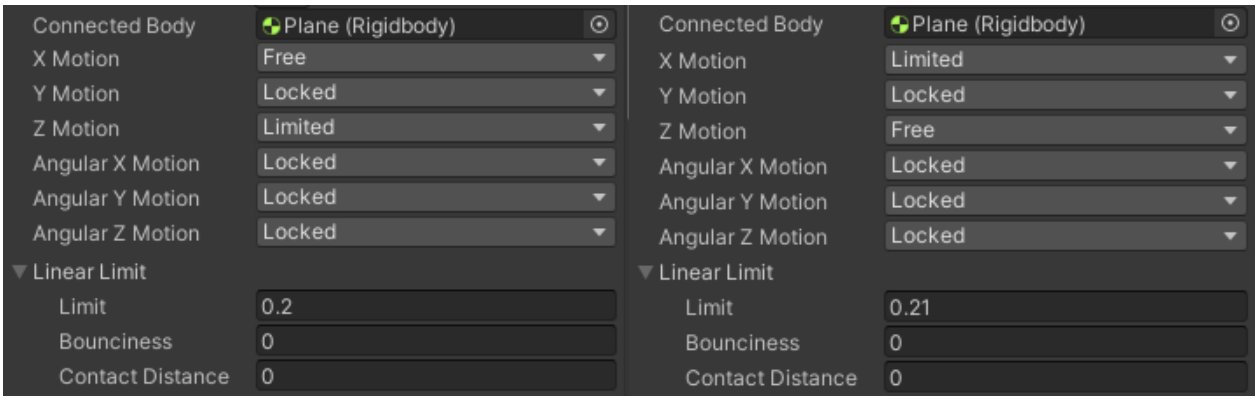

Figure 4.5: Configuration of parameters in Unity editor for a rectangular planar constraint

4.1.3 Hinge Constraint

The hinge constraint is a constraint that restricts all motion of an object except for rotation about a single axis. We implement this constraint using Unity's Hinge Joint script. Figure 4.7 shows a flat prism constrained to a folder under this constraint. The Hinge Joint has the key parameters of the Connected Body and Limit properties, with two additional key parameters:

- Anchor - Determines the position relative to the object about which it will rotate
- Axis - Determines the axis of rotation

Figure 4.6: Configuration of parameters in Unity editor for a hinge constraint

When configuring the hinge constraint, the Connected Body is also set to the marker object or one of its static children. The Anchor is set to the point of rotation, and the Axis is set to the desired axis of rotation. When limits are not set, the object is free to rotate in either direction 360 degrees about the Anchor. Using limits, the maximum and minimum angle of rotation is set relative to the initial position of the object. Figure 4.6 is a configuration of this constraint with limits.

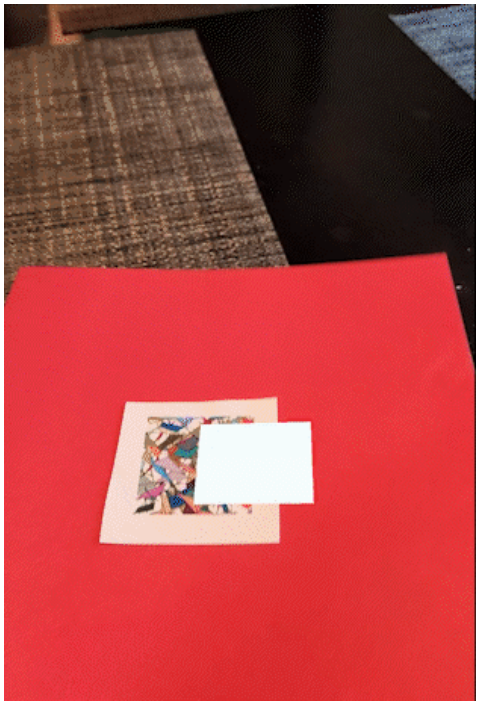

Figure 4.7: Flat prism with a hinge constraint anchored on prism edge

4.1.4 Combination constraint

The final type of constraint, the combination constraint, is not a standalone constraint like the three previously mentioned, instead it is any combination of the others. This combination of constraints allows for more complex constraints to be created. In this thesis we use a combination of constraints composed of slide and hinge joints to create a constraint with the behavior of a folding door. More details on an example of this constraint will be covered in the next chapter.

4.2 Handling Interaction

In this section we will cover how interaction between the user and the constrained objects is handled. There are two types of interactions supported: a push interaction and a grab interaction.

Of the two types of supported interactions, the push interaction is the simpler of the two. This interaction is handled by Unity using colliders. By attaching colliders to both the finger object and the constrained object, when the finger object contacts the constrained object, the object will move based on the constraint depending on the contact direction of the finger. If there is no collider attached to either the finger or the constrained object, no pushing interaction will occur. Figure 4.8 is an example of this interaction with a constrained cube being pushed.

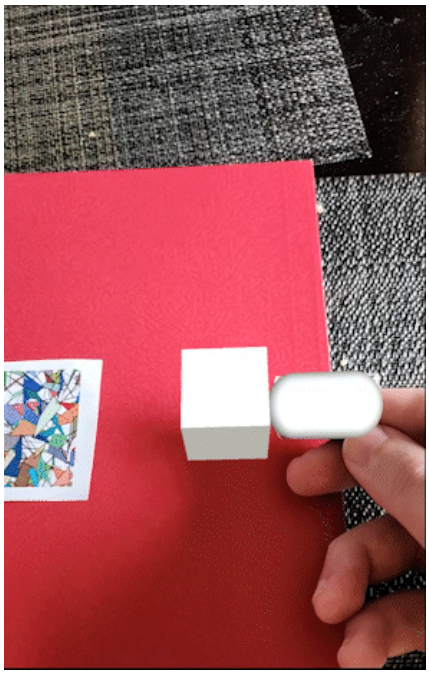

Figure 4.8: Cube with Slide constraint being moved with push interaction

4.2.1 Grab Interaction

The second type of interaction is the grabbing interaction where the virtual objects are grabbed by the user. Figure 4.9 shows this grabbing interaction on an object under a slide constraint. This interaction is more complex than the push interaction; to facilitate this interaction we implemented two scripts: the *FingerState* and the *GrabScript*. Along with these two scripts we also define a new type of object, an interaction object that is tagged as an *Interactable*. We define this interaction object to make a distinction between the grab interactions and push interactions. In this way you can attach an *Interactable* to an object to allow for the grab interaction while also allowing the rest of the object to facilitate the push interaction.

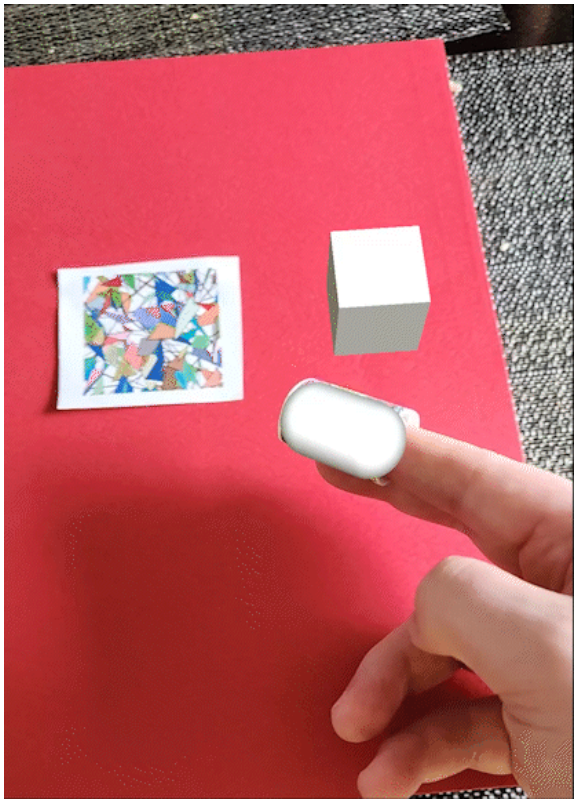

Figure 4.9: Cube with Slide constraint being moved under grab interaction

4.2.2 FingerState

The FingerState script is attached to the finger object. There are three main purposes of this script: the first is to keep track of the state of the finger, whether it is grabbing an object or not, the second is handling a grabbed object, and the third is releasing the object.

The first purpose of tracking the state is handled by a state variable called *emptyState,* when this variable is false the finger is not holding any object and is ready to interact with, when this variable is true this indicates that there is an object currently being grabbed and that another

object can not be grabbed until the current one is released. Figure 4.10 shows the code for setting a new object to grab.

```
public void SetInteract(GameObject newInteractObj)
{
    if(emptyState)
    {
        heldObj = newInteractObj;
        emptyState = false;
    }
}
```

Figure 4.10: Code for setting a new grabbed object

The second purpose is handling the object once it has been grabbed. This is done by updating the held object's velocity to move in the direction of the finger. Figure 4.11 shows the code for determining the direction of the grabbed object's velocity and updating it. In the case of grabbing a constrained object, this will cause it to follow the finger while still obeying the constraints that the object is under. The speed of this interaction is set by the user through the Unity editor interface.

```
void CheckRelativePos()
{
    dir = (trackedFinger.transform.position - heldObj.transform.position).normalized * interactionSpeed;
}

void UpdateInteractObjPos()
{
    heldObj.GetComponent<Rigidbody>().velocity = dir;
}
```

Figure 4.11: Code for updating the direction of the object's motion to follow the finger

The third purpose is to handle releasing the held object. When the object is released its velocity is set to zero and then the *emptyState* variable is set back to true, allowing a new object to be grabbed. An object is released when the user taps the screen of the AR device.

4.2.3 GrabScript

The GrabScript is also attached to the finger object, and its sole responsibility is to handle selecting the objects that will be grabbed by the finger. For an object to be grabbed by the GrabScript it must be tagged as an *Interactable* as well as have a collider that is set to be a trigger. Figure 4.12 shows the configuration of an interaction object in Unity. When the finger contacts the trigger collider it checks if the object is an *Interactable* and if the finger is currently holding an object. If both conditions are met, then the object is grabbed by the finger. Otherwise, the object will be ignored and instead if a non-trigger collider also exists then a push interaction will occur instead.

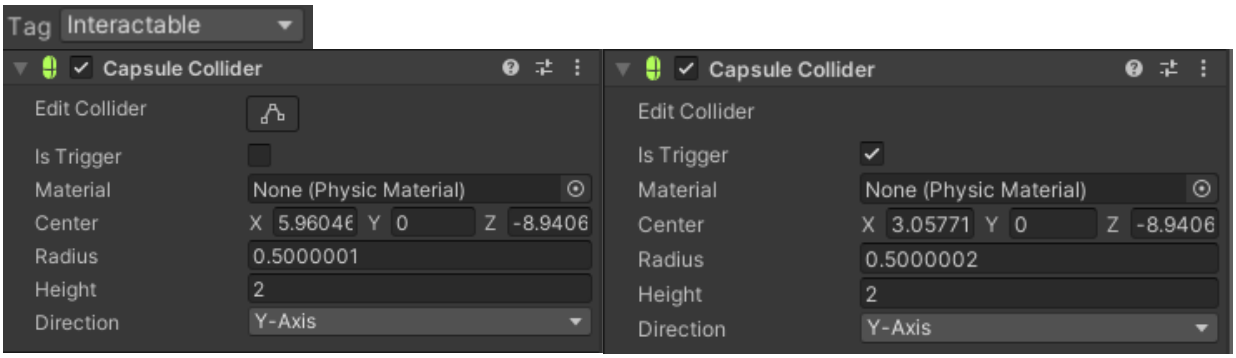

Figure 4.12: Configuration for an interaction object

4.3 Constrained AR Creation Process

This section will cover the process developed to create constrained AR experiences. Figure 4.13 breaks down the main steps in the process.

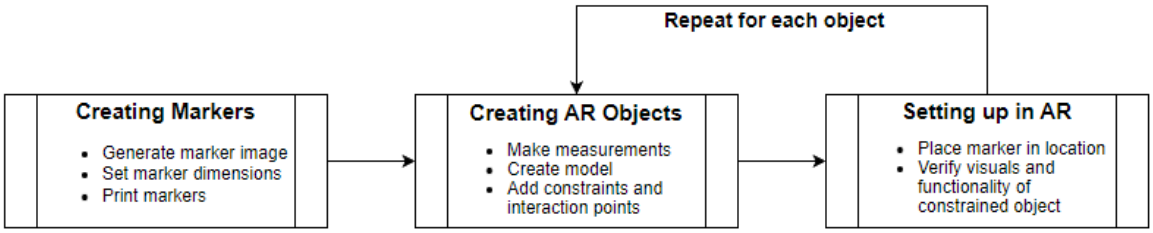

Figure 4.13: Diagram of AR experience creation process

4.3.1 Creating Markers

The first step in the process is to create the markers that will be used to display the AR objects as well as the finger that is used for interaction. This is a one-time step for each experience. For every object a marker image must be generated. Markers that are of high quality through rich details, good contrast, and avoiding repetitive patterns will result in better quality of tracking. Once the marker image is generated, the size of the physical marker needs to be determined and set in Vuforia's Target Manager. In the case of the cylindrical marker, measurements of the finger are needed for the marker to fit. Once the dimensions are determined, the markers need to be printed and constructed in the physical world.

4.3.2 Creating AR Objects

The next step in the process is to create the AR object that will be displayed onto the physical world. When creating the object, the physical location where the object will be placed must be determined. After the location is determined, measurements are made for the object to properly fit in the location. After measurements are taken, the actual model is created, either from scratch

using primitive shapes or using a pre-existing model. The model's dimensions are then adjusted to match the measurements made previously. Constraints are then added and configured to the model. To handle grabbing interaction, interaction points are added as either an actual object attached to the model, such as a handle, or as a location on the model itself.

4.3.3 Setting up in AR

Once a constrained object has been created, both the visuals and functionality need to be verified in the physical world. This process begins by placing the marker in the location determined earlier. The application is then run, first verifying to see if the model properly fits the location in terms of dimensions. If any adjustments are needed, return to Unity and make changes to the model, and check again to see if the model fits. This process may need to be repeated until the model fits. Similarly, once the model's visuals are taken care of, the functionality of the constraints are tested. This process is also done iteratively until the constraint is functioning as desired. When both the visuals and functionality are verified, the whole process is repeated for each object to be added to the experience.

Ch 5 Results

This chapter presents our results of applying the four types of constraints to create functional virtual objects based on commonly seen home decoration items. Additionally, we perform an evaluation of these constraints and the virtual by physical categorization using the criteria defined in Chapter 3.3.

5.1 Sliding constrained objects

This section covers the three pieces of decor we designed and implemented for the sliding constraint: a sliding cabinet door, a full-size multi-panel sliding door, and a drawer.

5.1.1 Cabinet sliding door

The cabinet sliding door is a two-piece door created for an open cubby on a rolling cabinet. The first piece is stationary and covers half of the opening of the cubby. The second piece is attached with a sliding constraint that constrains it along the base of the cubby, only allowing it to move the length of the cubby. Attached to the constrained piece of the sliding door is a handle which allows for a grab type interaction of the sliding door. Figure 5.1 shows the cubby with and without the sliding door and an interaction of opening and closing the sliding door. Note that the door slides smoothly and respects the limit of the rail length as the sliding door is stopped even as the finger continues beyond.

Figure 5.1: Rolling Cabinet empty and with the sliding door, and interaction of opening and closing door

5.1.2 Multi-panel sliding door

The multi-panel door was designed to be placed in the entranceway between two rooms. It has three panels. The first panel is a static panel that is located at the end next to the wall. The second panel is able to move, but is under a sliding constraint with the floor. This constraint only allows this panel to move between the wall that the first panel is next to and a full panel's length away from the wall. The third panel is similarly under a sliding constraint with the floor, with the limits of its constraints allowing it to move the full length of the entranceway. The second panel is set to follow the third panel, as the third panel moves out, the second panel will follow until it reaches the limits of its constraint. To allow for a grabbing interaction with the door, an interaction point is placed halfway up at the end of the third panel. Figure 5.2 shows the entranceway before and after adding the sliding door and the sliding door being closed by grabbing the interaction point. Compared to Figure 5.1 there is a considerable size difference between the two doors, yet the large door operates well while also working with multiple constraints at once.

Figure 5.2: Entranceway empty, with multi-panel door, and interaction closing door

5.1.3 Drawer

The drawer was created for the rolling cabinet cubby. The whole drawer is under a slide constraint, which restricts its movement between fully pulled open and pushed all the way in. Figure 5.3 shows the cubby with and without the drawer as well as the drawer being pulled open all the way. There is a handle in the middle of the front panel for grabbing the drawer.

Additionally, since the drawer goes inside a physical cubby, invisible planes act as the cubby walls to occlude the drawer when it is pushed in so that it can not be seen through the physical cabinet. In addition to the smooth motion seen in Figure 5.3, we also observe how intuitive the grabbing interaction is with the drawer.

Figure 5.3: Cubby empty and with drawer, and interaction pulling open drawer

As discussed, in all three cases, these three demonstrate fully functional objects with intuitive interactions and limits. In regard to the integration of these constrained objects, the drawer and the smaller sliding door blend well, especially the drawer due to the occlusion of the drawer when pushed all the way in. Whereas the large sliding door lacks in this regard as it sticks out in the open space.

5.2 Planar constrained objects

In this section we will go over the two planar constrained objects that were created: a chair constrained to the floor, and a hanging picture constrained to a wall.

5.2.1 Chair

This constrained chair was designed to go with a dinner room table set. The chair is under a rectangular planar constraint. The limits set on the constraint allow the chair to be pushed in and out from the table, as well as moving along the length of the table. Figure 5.4 demonstrates the chair being moved to the edge of its rectangular constraints. The interaction point is located at

the top of the chair's backrest. When examining the functionality and interaction of the chair, it closely resembles that of pulling a real chair out to sit on.

Figure 5.4: Table with before and after the chair is added, and demonstration of the chair pulled to constraint edges

5.2.2 Hanging Picture

The hanging picture is placed under a rectangular planar constraint. This constraint was set such that the horizontal movement spans to both ends of the wall, and the vertical movement is within one picture height up and down. Movement of the picture until it hits the lower limit can be seen in Figure 5.5. The interaction point for the picture is on the picture within the frame. When moving the picture, it glides nicely along the wall. It also properly obeys the limits of its constraints, stopping as it reaches the end in any direction.

Evaluating the planar constraint in terms of integration, both the chair and the hanging picture blend well with their surrounding environments. Furthermore, the chair's integration is seamless such that at first glance, it is difficult to tell the difference between it and a real chair around the table. Similarly, the hanging picture also integrates at a level that differentiation that it is fake is a challenge.

Figure 5.5: Wall without picture, with picture added, and moving wall picture to lower limit

5.3 Hinge constrained objects

For the Hinge constraint there are two objects that we created: a table extension and cabinet doors.

5.3.1 Table Extension

The table extension is constrained to the end of a table using a hinge constraint. The constraint allows the extension to hinge about the location where the extension and the table meet. The limits on the constraint have a lower limit perpendicular to the table surface when in the down-most position, and an upper limit flush with the table surface when in the up-most position. Located at the end opposite from where extension and table meet is the interaction point. Figure 5.6 shows the stages of creating the table extension and interaction with the extension. Interaction with the table extension is especially intuitive, due to the grab and push interactions being very straightforward for the extension. Despite being a larger object, the motion is very smooth when moving the extension.

Figure 5.6: Table without extension, with prototype extension, with completed extension, and extension being lifted up.

5.3.2 Cabinet Doors

The cabinet doors consist of two doors constrained with a hinge constraint to the open cubby space of a rolling cabinet. The hinge point on either piece is located at the ends of the cubby, allowing the doors to be opened from the middle outwards. The limits of rotation on both doors begin at the closed position to 105 degrees of rotation when fully opened. Each door has a handle that acts as an interaction point. Figure 5.7 displays the cubby empty, with the hinge doors, and one of the hinge doors being opened. The functionality of the hinge door presents itself nearly identical to opening a real cabinet door. Similar to the drawer, we can also see the intuitive grab interaction for the hinge door in Figure 5.7.

Figure 5.7: Empty Cubby, Cubby with hinge doors, and a hinge door being opened

Assessing the hinge constraint in terms of integration, the cabinet door blends with the rolling cabinet such that it appears as if it is an actual part of the cabinet. The table extension, while not to the point of the large sliding door, stands out to some degree. There were challenges with matching the texture of the extension to the table, so while it closely matches the table, they are not exactly the same.

5.4 Combination constrained object

For the combination constraint, we designed and constructed two sectioned folding doors. The folding doors were designed to fit in the open cubby space. Each door has two sections. On either door there is a handle for a grab interaction. Each folding door individually contains three constraints:

1) A hinge constraint on the outer section attached at the end of the cubby, allowing this section to hinge similar to the cabinet doors. The limits on this constraint are from the closed position flush with the opening, to 90 degrees of rotation, such that the section is perpendicular to the opening.

2) A hinge constraint on the inner section that connects the two sections of the folding door. The hinging anchor is located where the two sections meet. The limits for this constraint are that the sections are in line with one another when closed and stacked next to each other when fully opened.

3) A sliding constraint on the inner section of the door that constrains this section to the bottom of the cubby space. Unlike a normal sliding constraint used previously, there are two differences used in this one. First, where a standard sliding constraint would not allow for the constrained object to have any rotation, this constraint allows the door section to rotate about its vertical axis. Second, rather than the center of the section, the inner end of the section is the part that is allowed to slide. There are no limits for this constraint because the limits of other constraints restrict the range of this constraint. Figure 5.8 shows the cubby before and after adding the folding doors, and all three constraints working as the folding doors are opened.

Figure 5.8: Cubby without folding doors, with folding doors, and interaction opening doors

Evaluating this constraint, the functionality, interaction, limits, and integration all meet a satisfactory level. However, the key success of this constraint is in the ability to combine constraints together in a meaningful way.

5.5 Multiple constrained objects

In the previous sections we considered each constrained object on its own, in this section we have two scenarios that contain multiple constrained objects in a single scene. Both scenarios contain three objects with different constraints.

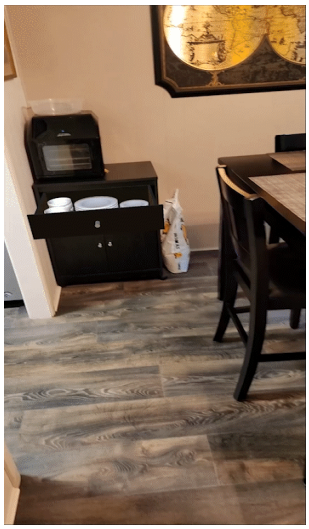

Figure 5.9: Drawer, extension, and chair in single scene

The first scenario, shown in Figure 5.9 consists of the planar-constrained chair, the hinge-constrained table extension, and the sliding-constrained drawer. The second scenario shown in Figure 5.10, consists of the sliding-constrained multi-section door, the planar-constrained hanging picture, and the combination-constrained folding doors. In each of these scenarios, the tracking mode for all the constrained objects is set to Extended Tracking, which allows all of the objects to be present simultaneously.

Figure 5.10: Folding doors, hanging picture, and multi-section sliding door in a single scene

The main focus for evaluation of these scenarios is in terms of scalability, as the individual constraints used within the scenarios were discussed in the previous sections. Both these

scenarios demonstrate the ability to support multiple constraints in a single scene. In either scenario the additional constraints were supported without the loss of functionality from any.

5.6 Evaluating results

When evaluating the different constraints, we found that they all properly represented to a reasonable degree the physical objects they were based upon. Following this, we are able to verify that the virtual by physical categorization is meaningful through these various applications. Additionally, we have also shown that this category of constraints can be used in validating functionality and visuals of real-life models.

Chapter 6 Conclusion

The purpose of this thesis was to investigate using virtual by physical constraint-based AR to validate functionality and visuals of real-life models. To accomplish this, we created directly interactable constrained objects that were placed in the physical world. To help examine these objects we used a home decoration environment as a way to integrate our constrained objects into the physical world.

In our initial investigation we were able to understand how to work in AR using Vuforia and Unity. We learned how markers are tracked and determined which tracking modes worked best for our needs. When examining support for direct interaction, we developed a method that allowed for intuitive interaction with constrained objects. Furthermore, we discovered limitations of the system, and how to mitigate them.

For our thesis, we used three types of constraints: the sliding constraint, the planar constraint, and the hinge constraint. To create these constraints, we configured Unity's built-in joints, and defined limits for each constraint. In addition to the three types of constraints, we were able to develop a functional constrained object using a combination of multiple different constraints. We created a process for creating constrained objects that we followed when creating all of our objects in this thesis.

For each constraint type, we created multiple functional constrained objects. The created constrained objects were integrated into the environment seamlessly, and interaction with the objects was intuitive and straightforward. Additionally, we were able to support multiple constrained objects in the same scene as well as multiple constraints on a single object. With these constrained objects we were able to verify that the virtual by physical categorization was meaningful and demonstrate their usefulness in validating functionality and visuals.

We categorized constrained AR based on object and interaction types and identified a well-defined AR application domain, virtual by physical, for focused investigation. This categorization allows for organizing existing results, such as the constraint type description from Wang et al. [7] and focused feasibility study from Moehring and Froehlich [17], under a unified framework for discussion. As a result, we have identified and demonstrated the potentials of virtual by physical constrained-based AR for validating functionality and visuals. As part of the prototype process, we have refined and formulated a template for efficient and effective configuring and constructing future virtual by physical investigations.

We defined a systematic process for creating virtual by physical constraint-based AR. The flow chart in Figure 4.13 (Section 4.3) identifies the key points of the process and serves as a template for creating virtual by physical AR applications. By following this process, we have successfully

created multiple versions of the augmented room, where, guided by the documented process, subsequent revisions and scene compositions were achieved with increasing efficiency and effectiveness.

6.1 Limitations and future work

As one of the main components of the thesis was to validate the visuals of real-life models, unintended occlusion of physical objects by virtual objects posed an issue. In many instances the hand which was used for interaction was covered by the virtual objects being interacted with, despite it being closer to the camera than the object. To improve on this, supporting occlusion by using depth sensing devices could help make the experiences appear more natural.

Our thesis uses sliding, planar, and hinge constraints as they represent typical constraints seen in life. However, even together with combinations of these constraints, they do not cover all of the possible constraints that exist in the real world. Our work could be expanded upon to support more types of constraints, such as a ball and socket constraint.

While we were able to accomplish an intuitive method for interaction that worked smoothly, it is also true that our method is very basic. One main restriction for this is the use of a smartphone as the AR device. Due to the handheld nature of the smartphone, only one hand is available when operating the applications. The second restriction is a limitation of using markers attached to the fingers. While Vuforia supports multiple marker detection capabilities, each additional marker reduces the overall tracking quality. Additionally, the occlusion of the fingers creates an additional issue that is non-trivial to work with. To help alleviate the first restriction, we could adopt the use of a hands-free method, either by mounting the phone to the head or using an existing head mounted display device. For the second restriction, although we chose to use a marker-based approach for the high accuracy, with the advancements of markerless approaches, switching could avoid these issues while maintaining the same performance.

As noted in Chapter 5, some constrained objects did not blend into the environment as well as most of the others did. In the case of the objects constrained in the cubby, the rolling cabinet provided features that helped them blend; where with the multi-paneled sliding door, it is located in a large open space alone. To help with integration, adding additional features such as a rail for the sliding door, could be a possibility.

References

[1] “Pokémon GO,” *Pokémon GO*. https://pokemongolive.com/ (accessed Apr. 17, 2023).

[2] A. Dünser, L. Walker, H. Horner, and D. Bentall, “Creating interactive physics education books with augmented reality,” in *Proceedings of the 24th Australian Computer-Human Interaction Conference*, in OzCHI ’12. New York, NY, USA: Association for Computing Machinery, Nov. 2012, pp. 107–114. doi: 10.1145/2414536.2414554.

[3] C. for D. and R. Health, “Augmented Reality and Virtual Reality in Medical Devices,” *FDA*, Dec. 2022, Accessed: Apr. 17, 2023. [Online]. Available: https://www.fda.gov/medical-devices/digital-health-center-excellence/augmented-reality-and-virtual-reality-medical-devices

[4] “IKEA launches IKEA Place a new app that allows people to virtually place furniture in their home.” https://about.ikea.com/en/newsroom/2017/09/12/ikea-launches-ikea-place-a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home (accessed Jan. 26, 2023).

[5] L. H. Hansen, P. Fleck, M. Stranner, D. Schmalstieg, and C. Arth, “Augmented Reality for Subsurface Utility Engineering, Revisited,” *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 11, pp. 4119–4128, Nov. 2021, doi: 10.1109/TVCG.2021.3106479.

[6] S. Nuanmeesri and P. Kadmateekarun, “Augmented Reality to Teach Human Heart Anatomy and Blood Flow,” *Turk. Online J. Educ. Technol.*, vol. 18, no. 1, 2019.

[7] T. Wang, X. Qian, F. He, X. Hu, Y. Cao, and K. Ramani, “GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications,” in *The 34th Annual ACM Symposium on User Interface Software and Technology*, in UIST ’21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 552–567. doi: 10.1145/3472749.3474769.

[8] T. Piumsomboon, D. Altimira, H. Kim, A. Clark, G. Lee, and M. Billinghurst, “Grasp-Shell vs gesture-speech: A comparison of direct and indirect natural interaction techniques in augmented reality,” in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Sep. 2014, pp. 73–82. doi: 10.1109/ISMAR.2014.6948411.

[9] P. Boonbrahm, C. Kaewrat, and S. Boonbrahm, “Interactive Augmented Reality: A New Approach for Collaborative Learning,” in *Learning and Collaboration Technologies*, P. Zaphiris and A. Ioannou, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 115–124. doi: 10.1007/978-3-319-39483-1_11.

[10] T. Tanikawa, H. Uzuka, T. Narumi, and M. Hirose, “Integrated view-input interaction method for mobile AR,” in *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, Mar. 2015, pp. 187–188. doi: 10.1109/3DUI.2015.7131763.

[11] “IVO - Interactive Virtual Object Engine For Augmented Reality And Virtual Reality Environments,” *Zugara*, Apr. 20, 2016. http://zugara.com/ivo-interactive-virtual-object-engine-for-augmented-reality-and-virtual-reality-environments (accessed Jan. 24, 2023).

[12] Z. Wang, S. Ong, and A. Nee, “Augmented reality aided interactive manual assembly design,” *Int. J. Adv. Manuf. Technol.*, vol. 69, no. 5–8, pp. 1311–1321, Nov. 2013, doi: 10.1007/s00170-013-5091-x.

[13] P. Boonbrahm and C. Kaewrat, “Assembly of the Virtual Model with Real Hands Using Augmented Reality Technology,” in *Virtual, Augmented and Mixed Reality. Designing and Developing Virtual and Augmented Environments*, R. Shumaker and S. Lackey, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 329–338. doi: 10.1007/978-3-319-07458-0_31.

[14] J. Patten and H. Ishii, “Mechanical constraints as computational constraints in tabletop tangible interfaces,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, in CHI ’07. New York, NY, USA: Association for Computing

Machinery, Apr. 2007, pp. 809–818. doi: 10.1145/1240624.1240746.
[15] S. An *et al.*, “ARShoe: Real-Time Augmented Reality Shoe Try-on System on Smartphones,” in *Proceedings of the 29th ACM International Conference on Multimedia*, Oct. 2021, pp. 1111–1119. doi: 10.1145/3474085.3481537.
[16] B. Nuernberger, E. Ofek, H. Benko, and A. D. Wilson, “SnapToReality: Aligning Augmented Reality to the Real World,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, in CHI ’16. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 1233–1244. doi: 10.1145/2858036.2858250.
[17] M. Moehring and B. Froehlich, “Natural Interaction Metaphors for Functional Validations of Virtual Car Models,” *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 9, pp. 1195–1208, Sep. 2011, doi: 10.1109/TVCG.2011.36.
[18] J. Cheng, K. Chen, and W. Chen, “Comparison of marker-based AR and markerless AR: A case study on indoor decoration system,” Jul. 2017. doi: 10.24928/JC3-2017/0231.
[19] “Vuforia Enterprise Augmented Reality (AR) Software | PTC.” https://www.ptc.com/en/products/vuforia (accessed Apr. 29, 2023).
[20] K. Cieślik and M. J. Łopatka, “Research on Speed and Acceleration of Hand Movements as Command Signals for Anthropomorphic Manipulators as a Master-Slave System,” *Appl. Sci.*, vol. 12, no. 8, Art. no. 8, Jan. 2022, doi: 10.3390/app12083863.
[21] M. C. F. Macedo and A. L. Apolinário, “Occlusion Handling in Augmented Reality: Past, Present and Future,” *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 2, pp. 1590–1609, Feb. 2023, doi: 10.1109/TVCG.2021.3117866.