Annexing Reality: Enabling Opportunistic Use of Everyday Objects as Tangible Proxies in Augmented Reality

Anuruddha Hettiarachchi, Daniel Wigdor
Department of Computer Science
University of Toronto
{anuruddha | daniel}@dgp.toronto.edu

Figure 1. Left: a user's table as seen un-aided by augmented reality. Right: The same user's table, as seen through our AR headset, augmented with our Annexing Reality tool. Note that virtual content is placed and scaled in order to opportunistically take advantage of physical objects in order to provide haptic experiences for virtual ones.

ABSTRACT

Advances in display and tracking technologies hold the promise of increasingly immersive augmented-reality experiences. Unfortunately, the on-demand generation of haptic experiences is lagging behind these advances in other feedback channels. We present Annexing Reality; a system that opportunistically annexes physical objects from a user's current physical environment to provide the best-available haptic sensation for virtual objects. It allows content creators to a priori specify haptic experiences that adapt to the user's current setting. The system continuously scans user's surrounding, selects physical objects that are similar to given virtual objects, and overlays the virtual models on to selected physical ones reducing the visual-haptic mismatch. We describe the developer's experience with the Annexing Reality system and the techniques utilized in realizing it. We also present results of a developer study that validates the usability and utility of our method of defining haptic experiences.

Author Keywords

Augmented reality; opportunistic tangible interfaces; augmented reality content authoring

CHI'16, May 07 - 12, 2016, San Jose, CA, USA

ACM 978-1-4503-3362-7/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858134

INTRODUCTION

The goal of Augmented Reality is to seamlessly blend real and virtual content, to create an alternative reality for a user comprised by a mix of the fantastical and the actual [29]. Thanks to today's advanced display, tracking and spatialized audio technologies, this is, in some sense, close to being realized [44]. The experience, however, is brittle: as soon as the user reaches out to touch a virtual object and finds only air, the illusion is spoiled. There is a clear need to meet the immersive capabilities of audio and visual output with equally immersive haptic experiences.

Considerable effort has been expended to address this deficiency. Techniques for providing haptic output include, mechanical-actuator based haptic feedback systems [1,28,33], air-jet driven force feedback devices [15,40,41], shape changing displays [19,22,27], string-based haptic feedback systems [13,37] and more recently electro-vibration [36] and laser-based systems [30]. While each of these shows considerable promise, each also has significant drawbacks. Expressivity, cost, the need to be attached to the user, the strength of the feedback, and other limitations continue to hinder their adoption. Moreover, touch sensation provided by these techniques is limited, and often lacks a host of other rich haptic details. In short, while seeming promisingly close, we are in fact far from being able to reach-out and touch virtual objects.

One proposed solution is to use everyday objects to physically represent virtual ones [18,20]; however, it is impractical to find a physical prop from any given space that is identical to a virtual object. It is also unrealistic to require

a user to purchase a custom set of props for each virtual experience. Thus, a mismatch between the virtual and physical objects is inevitable. Simeone et al. [39] have studied how this mismatch affects user's feeling of immersion and engagement and have found that greater the degree of mismatch, lower the believability of the experience. This confirms the findings of Kwon et al. [26] on the effect of shape and size of physical props: object manipulation is more efficient when physical and virtual objects are alike in shape and size. Hence, selecting a physical prop that is as similar as possible to a virtual object is crucial for the user's suspension of disbelief.

In this paper, we present *Annexing Reality*; a system that opportunistically finds physical objects from the user's surrounding that would best represent a given set of virtual objects, and then tailors the virtual objects to match the shape and size of the annexed physical ones. In so doing, we are able to minimize the mismatch between the virtual object and the physical prop to which it is mapped. This enables the development of arbitrary models, rather than forcing designs to match a set of known props. On-demand mapping to available props makes the system portable, providing the user the freedom to grab an AR headset and head out the door, rather than having to carry a bag of props.

We describe a developer experience in which the creator of a virtual object matches its shape to a set of basic primitive shapes, similar to a collision model. Additionally, tolerances for deformation of the virtual object are specified, as are preferences for matching (e.g.: it is more important to match a cylinder than it is to get the size right). The system then uses this information to optimally match virtual and physical objects in the user's environment and performs 3-DOF scaling on the model to improve physical correspondence.

We devise the goal of matching virtual and physical objects as a combinatorial optimization. The *Annexing Reality* system scans the user's physical environment with a Kinect depth sensor [45], discovers available physical objects, and derives their critical primitive shape and size information. It then casts a vote for each virtual-physical pair based on the shape and size similarity between the two. The voting algorithm allows for a three-level priority scheme where level-1 defines matching priorities for individual virtual objects, level-2 defines matching priorities for different primitive parts of each virtual object, and level-3 defines matching priorities for physical dimensions of each primitive part. Once the voting phase is complete, our system uses Hungarian algorithm [25] to find the optimal assignment that maximizes the total vote. Finally, virtual objects are overlaid atop assigned physical props in real-time allowing the user to tangibly interact with the virtual (Figure 1); the system rescales virtual objects to fit matched physical props, further reducing the mismatch.

In this paper, we present the following contributions:

1. *The Annexing Reality method for parametrically matching and adjusting virtual objects to physical ones*
2. *The developer UI and method for specifying preferences for attributes in matching and scaling*
3. *The results of a developer study which demonstrate the effectiveness of our methods and UI*

RELATED WORK

The Annexing Reality system builds upon four main areas of related research: visuo-haptic mixed reality, opportunistic tangible proxies, mismatch between the virtual and the real in augmented reality, and tools for augmented reality content authoring.

Visuo-Haptic Mixed Reality

Milgram et al. [29] formally defined Mixed Reality systems in which completely real environments are connected to completely virtual environments. VHMR allows the user to both see and touch virtual objects in a co-located manner such that the interaction becomes closer to reality [42]. Several systems have explored this idea of merging visual and kinesthetic perceptions in the virtual space, especially in medicine and surgery. In their early work, Carlin et al. [9] used VHMR for the treatment of spider phobia where a subject physically interacted with a virtual spider. Kotranza et al. [24] studied interpersonal touch and social engagement while interacting with a Mixed Reality Human: a mannequin with a virtual human overlay. Volumetric object models, derived from a CT, were used in conjunction with a PHANTOM 1.0 haptic feedback device [28] to simulate temporal bone surgery [2,32].

Among a variety of other areas, VHMR has also been explored in gaming and product design. Oda et al. [31] have developed a racing game in which the virtual car is controlled with a passive tangible prop. In the two-player ping-pong game developed by Knoerlein et al. [23], players can feel the impact of the virtual ball on the virtual bat with a co-located PHANTOM haptic device. ClonAR [12] developed by Csongei et al. allows product designers to physically edit 3D scans of real-world objects.

Opportunistic Tangible Proxies

Tangible proxies take advantage of humans' ability to grasp and manipulate physical objects as a means of interacting with digital information [21]. Instead of tightly coupling digital functionality to physical objects, OTPs adapt to the user's environment and leverage otherwise unused objects as tangible props.

In Opportunistic Controls [16], Henderson et al. suggests using physical affordances already present in the domain environment to provide haptic feedback on user inputs in augmented reality. Smarter Objects [17] associates a graphical interface with a physical object in a co-located manner allowing a user to provide tangible inputs while visualizing their effects. Cheng et al.[10], Corsten et al. [11],

and Funk et al. [14] propose repurposing everyday objects as tangible input devices. Cheng et al. place fiducial markers on everyday objects to convert them to input controls while Corsten et al. and Funk at al. use a Kinect depth sensor for object recognition and tracking. All of these require the user to create physical-digital pairs by programming physical objects. Alternatively, we focus on letting the system create physical-virtual pairs based on a developer's specifications.

Virtual-Real Mismatch

Since it is nearly impossible to find an exact physical replica of each virtual object within one's surroundings, there exists an inherent mismatch between characteristics of virtual objects and their physical props. In their recent work, Simeone et al. [39] have found that some amount of mismatch is acceptable especially where interaction between the user and the object is less likely to happen. However, increased mismatch negatively affects the user's believability of the virtual experience. From a similar study, Kwon et al. [26] have concluded that lower disparity results in greater realism and easier manipulation of virtual objects. With Annexing Reality, we intend to minimize the mismatch by dynamically finding a physical object from the user's surrounding that is as similar as possible to a given virtual object, and scale virtual objects to match the physical one's.

An interesting observation from the Pseudo-haptics literature is that, when visual and kinesthetic stimuli are fused together, visual stimuli dominate over kinesthetic stimuli resulting in illusionary haptic sensations. In BurnAR [43], subjects have reported involuntary heat sensation in their hands when virtual flames and smoke were overlaid onto their hands. In their series of visuo-haptic experiments, Ban et al. modified the curvature [5], angle of edges [4], and position of edges [6] of virtual shapes, without changing the physical prop which those virtual shapes were overlaid onto. For each virtual shape, the participants reported that they felt they were touching a physical shape similar to the virtual one being displayed. Based on these observations, we postulate that it is the critical shape that matters, thus we use geometric primitives to represent more complex shapes.

Augmented Reality Content Authoring

AR content authoring tools provide software frameworks on which developers can quickly create augmented reality applications without worrying about the functionality of underlying components. Such early systems [46,47] supported marker based virtual-real pairing where virtual objects were tied to unique image patterns known as fiducials markers.More recent commercial toolkits such as Vuforia [48] and metaio [49] enable use of physical objects as fiducials. In these systems however, the developer hard-pairs virtual objects to physical ones at the time of development, requiring the user to have the exact fiducials specified by the developer in order to use the applications. Taking a different approach, our Annexing Reality system scans and dynamically matches available physical objects from the user's physical environment to virtual ones.

DEVELOPER EXPERIENCE

The process begins with a developer creating a virtual scene, such as in an augmented reality game. An early goal of our project was developer transparency; that is, enabling matching and adjusting of virtual models to physical objects without any input from developers. We soon realized, however, that while this would provide a lower workload for our target users, it would also serve to disempower them in defining their intended experiences, as our tool would, in effect, be making design decisions on the fly. We thus turned our attention to defining a developer experience that is simultaneously powerful, easily grasped, and requires little work. To this end, we defined three guiding principles to guide our design of a tool to provide that experience:

1. *Do not impose any limitations on the properties of the models of the virtual objects that the users will see.*
2. *Provide a separation between the visual and physical properties to allow developers to focus on each individually.*
3. *Provide flexibility to enable the developers to prioritize any aspect of the physical properties of the objects to be matched.*

Annexing Reality Tool

The developer creates the virtual content using her preferred 3D modeling tools. Once complete, it is imported into our software, where she specifies which virtual objects should be matched to physical ones in the user's setting and which should not. In this manner, the developer controls which virtual objects are physically manipulated by the user. Other virtual objects may react to the behavior of user manipulated objects (e.g., the ball in a ping-pong game where the paddle is physically controlled by a player) or solely be controlled by the developer at the time of development (e.g. imagery in the background of the world).

For each user-manipulable virtual object, the developer generates a haptic model (e.g. Figure 2a) similar to a collision model. This begins with matching basic geometric primitives to the virtual object, which later on will be matched to the physical object(s) from the user's surrounding. The shape, size, and position of the model with respect to the virtual object reflect the haptic experience intended by the developer. As an example, consider the virtual model of a Champaign bottle, shown in Figure 2a. If the developer intends for the user to grab the bottle by the body, she would remove the conic part of the haptic model, and the bottle would be matched to cylindrical physical objects at its body. If her experience would be successful no matter which part which part of the bottle can be grabbed, she can leave the cylinder and cone in the model, and one or the other will be matched, with priority based on the weighting she will later define.

The next step is setting the parameters for the matching algorithm (Figure 2c). This requires the developer to order her *virtual objects* by priority (for situations where there is no sufficient number of physical objects for matching, or where one physical object equally matches to multiple virtual objects). Next, the developer prioritizes the *primitives* within

each of her haptic models (for situations where more than one of those primitives might be matched. As an example, with the Champaign bottle, the developer might prefer to match the body to a cylinder, and as a second choice to match the neck to a conic physical object). Finally, *physical dimensions* of those primitives are given preferences. Continuing to use the Champaign bottle as an example, the developer might find it more important that the matched cylindrical object have a small enough radius to enable grasping, and less important that its height precisely match that of the bottle. As the final step of haptic model generation, the developer sets maximum and minimum scaling limits for each dimension to avoid unnecessary rescaling of virtual objects.

Once the developer is finished creating haptic models, the system converts and saves this representation into a textual form which is understandable to the ObjectFinder module (see Figure 4) described in detail below, which is responsible for matching virtual and physical objects at runtime, parameterized by the properties set by the developer.

The reduction of the virtual model to a set of primitive shapes (the haptic model) is what gives Annexing Reality its power: the developer is provided with the opportunity to specify the haptic experience she wishes to provide, at a level of abstraction that allows, at development time, the tweaking of how 'perfect' a match must be in order to perform the annexation, or leave the object as virtual-only.

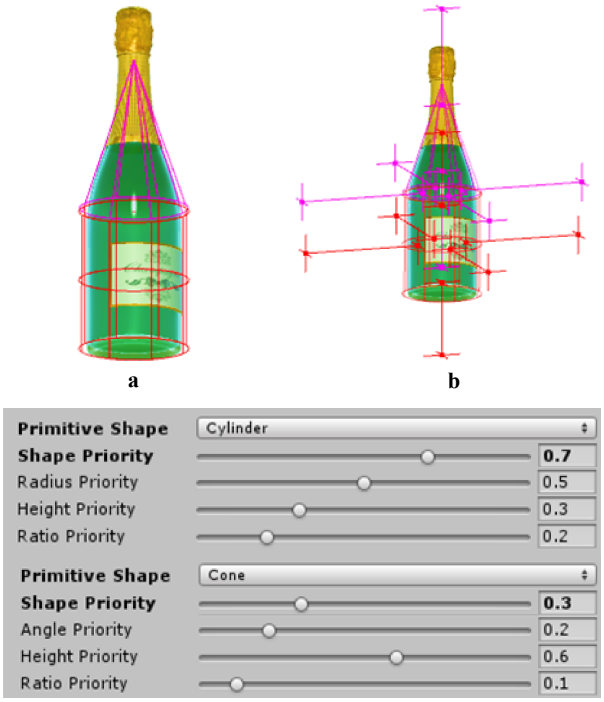

Figure 2. The developer UI for Annexing Reality. (a) The primitives of the haptic model, shown in wire frame, for a Champaign bottle object. (b) The developer sets min and max sizes for scaling each primitive (c) Ranking UI to prioritize matching.

OPPORTUNISTIC ANNEXING OF PHYSICAL OBJECTS

The haptic model forms the basis for matching performed at runtime. When virtual objects are added to a scene, the Annexing Reality system finds suitable physical objects as props from the user's surrounding and overlays corresponding virtual objects atop. The complete pipeline is illustrated in Figure 4. The ObjectFinder module is responsible for finding physical objects that are as similar as possible to virtual objects added by the developer. It uses a 3D map of the surrounding, retrieved from the FrameGrabber, to detect available physical objects, derive their salient primitive shape and size, and assign them to virtual objects based on the input from the HapticModelGenerator.

Object Shape Recognition

The main function of ObjectShapeRecognizer is to identify physical objects present in the Kinect's field of view. It continuously scans the user's physical environment (through FrameGrabber), detects horizontal planer surfaces (e.g. tables, floor), extracts objects lying on them, and then estimates their size, salient primitive shape, and orientation.

Object Cluster Extraction – The Annexing Reality system only matches physical objects that are supported by horizontal planer surfaces (see Limitations). As the first step, it detects and segments the dominant plane in the point cloud generated from Kinect's depth map. It then computes the 2D convex hull of the segmented plane. Next, a polygonal prism is created with the 2D convex hull as the base, which encapsulates the physical objects lying on the plane. Finally, it separates point clusters belonging to different objects using Euclidean Cluster Extraction method [35]. As a means of improving the accuracy of future shape detection phase, the Annexing Reality system reconstructs object models by smoothing and resampling point clouds using a Moving Least Squares method [3] (Figure 5a).

Shape Recognition – A RANSAC based shape detection method developed by Schnabel et al. [38] is used to recognize the dominant primitive shape (sphere, cylinder, cone, and torus) of each point cluster.

Voting

Once the salient primitive shape and the size of all point clusters are determined, each virtual object with no physical prop previously allotted votes for point clusters based on their shape and size similarity. The purpose of voting is to determine the optimal assignment of physical objects in which all virtual objects get reasonably similar physical props. The system applies the priority scheme set by the developer to weight the votes.

Virtual-Real Object Similarity Each virtual object is matched to each physical one and compared. If the physical object does not contain the primitive(s) of the virtual object's haptic model, the objects are considered dissimilar and no further processing is done on the pair. If one or more primitives is found within the physical object, the scale of each primitive is compared to its size in the haptic model.

A vote reduction method is used to determine the vote values. Thus, instead of calculating the size similarity, the Voter module calculates the opposite; the size mismatch. For a given dimension h (e.g. height, radius, or angle), the size mismatch Δ_h between a virtual object v and a physical object p is calculated as;

$$\Delta_H = \frac{\min(h_v,\ h_p)}{\max(h_v,\ h_p)}$$

The error ratio is taken instead of the difference to eliminate the influence of unit of measurement on the mismatch value.

Voting Equation – For a given virtual-physical object pair, the Voter iterates through all the primitive shapes of the virtual object (in the haptic model) specified by the developer, in the order of priority, to determine the shared primitive shape with highest priority value. If the search fails (there are no shared primitives), it assigns 0 as the vote given by the virtual object to that physical object. If a shared primitive is found, it starts the vote reduction process by first assigning the highest possible vote (a constant value C) and then multiplying that value by virtual object priority w_o and primitive shape priority w_s; each priority being a value between 0 and 1. To obtain the final vote value X_{vp} the Voter multiplies the already reduced vote by the sum of weighted dimensional mismatch values (D denotes the set of all dimensions for a given shape while w_h denotes priority given for the dimension h). The complete voting equation (vote cast by virtual object v on physical object p) can be expressed as:

$$x_{vp} = C \times w_o \times w_s \times \sum_{h \in D} \left(\frac{\min(h_v,\ h_p)}{\max(h_v,\ h_p)} \times w_h \right)$$

Matching

Once the voting is complete for each virtual-physical pair, the final step is to make the final match between physical and virtual objects. The goal is to maximize satisfaction with the match, based on the score provided by voting. As can be seen in Figure 3, some virtual objects, such as $v1$, have some degree of satisfaction from more than one match (55% satisfaction with physical object $p1$, and 34% satisfaction with $p2$). Other objects, such as $v2$, require a particular

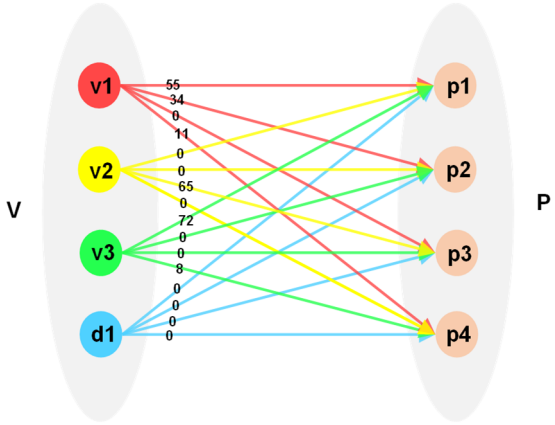

Figure 3. A weighted bipartite graph representing votes casted by virtual objects (V) on physical objects (P). A dummy node (d1) is added to make the graph symmetric.

physical object in order to be satisfied; in this case, 65% satisfaction with $p3$. In many instances, virtual objects may not get their first choice; in the scene described in Figure 3, $v1$ will be assigned to $p2$, so that $v3$ can be assigned to $p1$, thus maximizing the overall match score.

This matching task is formulated as a combinatorial optimization problem. The voting results in a weighted bipartite graph $G = (V, P; X)$ with the set of given virtual objects V and the set of candidate physical objects P as two partitions of G and each edge in X having a nonnegative weight (vote given by virtual object on the physical object connected by the edge). We add dummy vertices (dummy virtual objects if $n(V) < n(P)$ or dummy physical objects otherwise) and dummy edges with weight 0 to convert G to a symmetric weighted bipartite graph (Figure 3).

A *matching* M is a subset of edge set X such that each vertex is incident upon at most one edge in M. Our task is to find a *perfect matching* M^* (a matching in which every vertex in G is incident upon one and only one edge) such that the total weight of M^* is maximized. We use an open source C++ implementation [50] of Hungarian algorithm (Kuhn–Munkres method) [25] to solve this assignment problem. Once matching is complete, all point clouds are saved in order to be used as source models for object tracking.

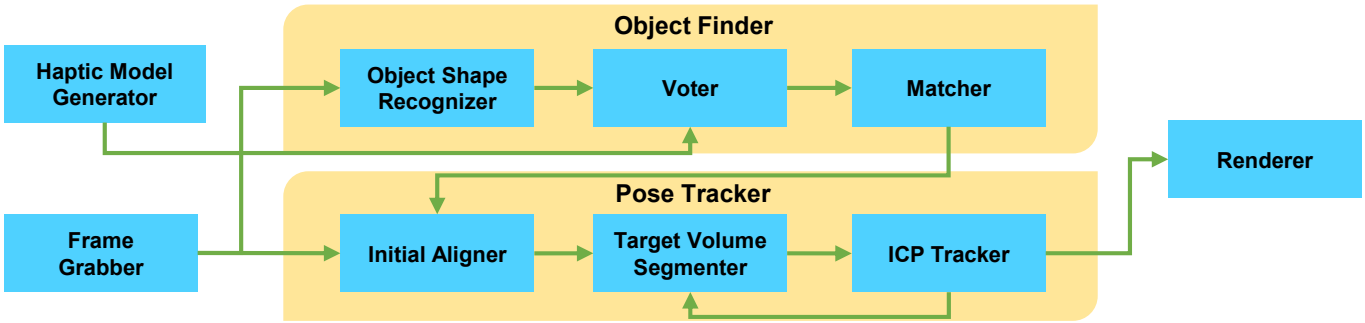

Figure 4. A dataflow diagram for the Annexing Reality system. Input to the system comes from the Haptic Model Generator tool (generated by the developer), and from the Frame Grabber module which reads from the Kinect Sensor.

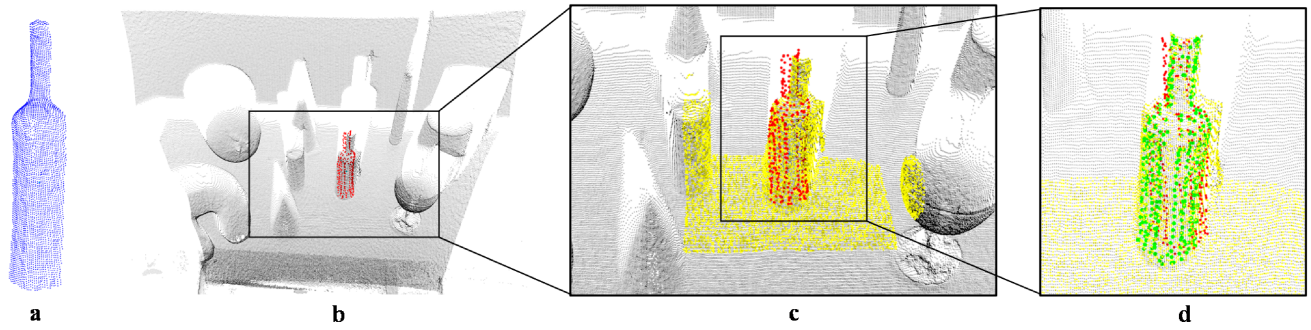

Figure 5. Object pose tracking pipeline, after the Object Shape Recognizer has identified an object for tracking: (a) reconstructed point cloud model of an object to be tracked (b) InitialAligner performs a rough pose estimation to locate the object in the field of view (red). (c) A cubic volume of the scene's point cloud (yellow) is segmented such that it encapsulates the object (d) IcpTracker computes the accurate pose (green) by incrementally aligning the reconstructed model to the point cloud inside the cubic volume.

OBJECT TRACKING AND RENDERING

Once matched, the Annexing Reality system overlays virtual objects on to physical props in real-time. The PoseTracker is responsible for tracking matched physical objects in Kinect's field of view while the Renderer renders virtual objects in place such that the user sees them atop props through the Head Mounted Display (HMD). Each virtual object has a separate tracking pipeline that tracks its matched physical object with following concurrent processes: InitialAligner, TargetVolumeSegmenter, and IcpTracker.

Given a source point cloud (Figure 5a) of a matched physical object, the InitialAligner searches for that object in the Kinect's field of view and estimates its rough pose if present (Figure 5b). It uses a RANSAC based pose estimation algorithm [8] with FPFH descriptors [34]. FPFH captures local (radius 2.5cm) geometric features of query points. The pose estimation algorithm uses these descriptors for feature matching. It eliminates poses that are likely to be wrong; a similarity threshold of 90% is used. The estimated pose acts as the initial guess to the IcpTracker.

As a preliminary step, the TargetVolumeSegmenter segments a cubic volume from the scene point cloud such that it encapsulates the 3D points of the target object (Figure 5c). The IcpTracker then attempts to refine the initial pose received from the InitialAligner by incrementally aligning the source point cloud to the object inside the target volume (Figure 5d). It uses Iterative Closest Point (ICP) algorithm [7] for this task. If the IcpTracker fails to find a satisfactory pose for the current frame, it rolls back to results from the previous frame and reattempts on the next frame. If the tracker fails for five consecutive frames, it restarts tracking with a new initial pose.

The Renderer retrieves poses of physical objects from the IcpTracker, derives poses of their primitives, transforms them from Kinect's coordinates frame to HMD's coordinate frame, scales and positions virtual objects in the scene based on size and pose of primitives, and renders a stereogram of the virtual scene on the HMD.

DEVELOPER STUDY

The Annexing Reality system aims to ease the development of adaptive haptic experiences for virtual scenes in AR. Although it is tempting to evaluate the power of bipartite matching in several real-world settings with different physical objects, we believe worthless if developers are not able to grasp the development process. Thus, we conducted a study to evaluate the utility and usability of the Annexing Reality system for developers. Our goals were twofold; the first was to understand how useful the set of information required of the developer to produce each haptic model (primitive shapes, scaling and preference information) is in producing good matches to physical objects. The second goal was to study developers' understanding and use of the UI (see Figure 2) for defining their desired haptic model.

Specifically, we set-out to answer the following research questions:

1. *Are developers able to express their design intent as a Haptic Model: a set of primitive shapes, priorities, and scaling parameters?*
2. *How usable is the Annexing Reality software tool for building that haptic model?*

Participants

Eight professional game developers and 3D modellers were recruited, all male, age 27-48 years, to act as expert assessors of our tool. All of them had both game development and 3D modeling experience. All had experience with at least one augmented reality application while three of them had developed augmented reality applications. Each participant received $50 for one hour session.

Apparatus

We used seven royalty-free 3D models found online as our virtual objects, which were to be matched to one or more of eight physical props (see Figure 6). The participants were used a Windows PC with a 24 inch monitor. At several points during the study, they wore a helmet-mounted Kinect with a Meta 1 head-mounted display [51] while testing and interacting with physical props (see Figure 1).

Task

The task was performed 4 times per participant – twice with simple virtual models with a single matching primitive shape (e.g. a lightsaber with a cylinder matched to the handle; see Figure 6), and then twice with more complex models which contained multiple primitive shapes (e.g. a lamp with a spherical body and cylindrical shade; see Figure 6). Each task was divided into two parts. First, the participants were shown one of the virtual models, and asked to imagine physical objects which they might want to match that virtual model to. They were then asked to use our Annexing Reality tool to construct the haptic model, including primitive shapes, prioritizations, and size characteristics to enable the desired matching. Next, the participants were given the set of physical props, and asked to group them into two sets: those which they expected to match their representation to, and those they did not. They were then asked to sort the props in the "will match" group to their expected matching probability: object with highest, second highest, third highest etc. probability of being selected by the system based on the haptic model they defined. Once each task was complete, the participants were given the opportunity to don the HMD to observe the effects of matches.

Procedure

Assessors completed a demographic questionnaire prior to the study. They were given the background to the project and a detailed explanation of the task, followed by a demonstration. Then, the participants completed the task four times: twice for basic, single-primitive virtual models, and then twice for more complex virtual models.

During each trial, we measured the time required to create a complete haptic model. We also recorded the frequency and type of errors participants committed. After finishing all four trials, the participants completed a post-study questionnaire. We asked them to rate their impression on learnability, efficiency, and satisfaction of the tool using a 5-point Likert scale. In addition, participants gave their expert opinion on utilizing geometric primitives to represent complex shapes and using a priority scheme to control matching.

Results

We collected data for 32 trials. After the short demo, participants were able to learn the tool and generate haptic models for the given virtual objects. On average, they took 2 minutes and 5 seconds to complete a single-primitive haptic model and 3 minutes 18 seconds to complete a two-primitive model. Participants committed errors while using the tool; on average 0.53 per trial. Some of the common errors were: accidently modifying other haptic models instead of the intended one, forgetting to set certain priority values and minimum-maximum scaling limits, and placing primitives in incorrect orientations.

Understanding of Haptic Models

Participants were successful in using haptic models to define the sort of physical objects that should be matched to virtual models. In all 32 trials, they successfully predicted the "will match" group: the set of hand-picked physical objects included the prop selected by the system. Interestingly, in only 19/32 trails, the participants successfully identified the object with the highest probability of being annexed by the system. Of the remaining trials, their second ranked object was chosen 10 times, and third ranked object 3 times. In all of these 13 trials, a two-primitive haptic model included nearly equal matching priorities for both primitive shapes. This suggests that a forced-rank system might provide a better UI to aid developers in understanding the matches which will be made with their haptic models.

Participants accepted the use of simple geometric primitives to represent more complex virtual models (median score of 4.5). They also expressed the necessity of using a priority scheme to control matching (median score of 4). However, participants disagreed that it was easy to precisely predict the outcome of the weighting without having the knowledge of internal vote calculation steps (median score of 3), although, in many cases (19/32), they successfully guessed the physical objects matched by the system. This indicates the necessity of some level of abstraction in UI for priority input or some mapping between more intuitive constructs that a user can relate to and metrics that serve as input to the voting mechanism.

Tool Usability

We asked participants to assess the Annexing Reality graphical tool on the following: the efforts needed to learn how to create haptic models using different functionalities of the tool; the ability to perform a task using the tool without leading to erroneous behaviours; and their satisfaction with the overall performance of the tool. Participants strongly agreed that the software was easy to learn (median score of 5). Many believed that the tool worked as intended without leading to errors (median score of 4) although those developers who had no previous experience with 3D modeling found that aspect of the system challenging. In general, participants were pleased with the performance (median score of 4).

To summarize the key findings: the Annexing Reality graphical tool was rated by the expert assessors as easy to learn and use. They could generate haptic models quickly, and only committed a very few errors. All assessors were satisfied with the overall performance of the tool. They were able to generate haptic models that matched given virtual objects to expected physical objects. However, further research is required to improve the predictability of the weighting scheme.

Figure 6: Top: the visual models used in our developer studies, overlaid with primitives from possible haptic models. Bottom: various visual models matched to physical objects based on the haptic models' match for physical characteristics.

LIMITATIONS

A primary limitation of the Annexing Reality graphical tool was made clear from the developer study: the difficulty of precisely predicting a matching physical prop based on the priority scheme of the haptic model. Although the goal of the system is to define a class of physical objects for annexation, which developers were able to do without trouble, their high rate of error in ranking objects within that class suggests either an incomplete understanding of the haptic models or of how they are used to perform matching. In particular, the priority scheme involves multilevel weighting in a continuous scale (0-1), and this prioritization would seem to not have been understood.

In terms of performance: object detection, shape recognition, and motion tracking limit the current implementation of the Annexing Reality system. The system can only detect physical objects supported by horizontal planer surfaces as it uses a plane segmentation based method for object cluster extraction. At the same time, due to erroneous depth information generated by the Kinect depth sensor, the system cannot detect transparent or glossy objects. Our ObjectShapeRecognizer module limits itself to recognizing only the most salient primitive shape of an object due to the fact that, inferring less salient shape information from imperfect, partial point clouds is error prone. Also, objects smaller than 7 cm × 5cm × 5cm do not generate sufficient number of 3D points in Kinect data to accurately determine their shapes, hence are not detectable in the current implementation. As the user wears the Kinect depth sensor on a helmet, quick head movements result in erroneous tracking due to high relative motion of objects with respect to Kinect. Even though we have not made a contribution to computer vision, we hope that our work will continue to motivate improvements.

CONCLUSION & FUTURE DIRECTIONS

We have presented Annexing Reality; opportunistic annexation of physical objects from the user's surrounding to temporarily provide best available haptic sensation for virtual objects in augmented reality. It allows users to freely move between environments while experiencing tangible augmented reality, without carrying special equipment or props with them. With our system, developers can define, a priori, adaptive haptic experiences for virtual scenes. Based on this specification, the system dynamically finds physical objects from the user's surrounding that are similar to virtual ones, and uses them to physically represent them. A developer study revealed that content creators accepted the Annexing Reality system as a useful tool for designing augmented reality applications in which virtual objects are opportunistically paired to and controlled by everyday physical objects. We identify numerous ways the system can be extended to further improve its usability and utility.

In future, we hope to improve developers' understanding of the priority scheme by providing tools for simulating voting and matching. In particular, providing a database of physical objects commonly found in a user's surrounding would allow for useful simulation during the development. Modeling grasp will help developers create more effective haptic models. In addition, we intend to simplify the prioritization mechanism, by replacing the continuous scale with a forced-choice ranking.

Although the set of primitive shapes currently implemented in our system allowed us to thoroughly explore the concept of Annexing Reality, adding more geometric primitives such as planes, cubes, and pyramids will be essential in providing more power to the developers when defining haptic experiences. We have begun exploration of advanced computer vision techniques to allow the detection of

additional features of physical objects such as texture, and applying object recognition technologies to understand weight and other meta data of objects seen by our system. We have also begun the augmentation of our apparatus to include a thermal camera which would allow us to use temperature as an additional matching parameter (and perhaps to avoid matches which could result in injury). The application of improved sensing and computer vision techniques will also enable the detection of more than one primitive shape in physical objects, allowing us to match more complex physical objects.

For an enhanced user experience, accurate, low latency tracking is crucial. As such, utilizing RGB texture information may help tracking rotations of geometrically symmetric objects. Moreover, being able to detect and track transparent, glossy, and deformable objects will expand the variety of objects that can be used with our system.

While haptic feedback technologies continue to improve, we believe that there is a clear opportunity for an immediate leap forward through annexation of physical experiences. We believe that this paper will serve as a guidepost to begin to overcome significant challenges of authoring experiences given a lack of information about the physical objects users will actually be interacting with.

ACKNOWLEDGEMENTS

We thank Bruno De Araujo, Katie Barker, Khai Truong, Jonathan Deber, John Hancock, and members of the DGP for their assistance in various stages of this work.

REFERENCES

1. Merwan Achibet, Maud Marchal, Ferran Argelaguet, and Anatole Lecuyer. 2014. The Virtual Mitten: A novel interaction paradigm for visuo-haptic manipulation of objects using grip force. *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, IEEE, 59–66. http://doi.org/10.1109/3DUI.2014.6798843

2. Marco Agus, Andrea Giachetti, Enrico Gobbetti, Gianluigi Zanetti, and Antonio Zorcolo. 2014. A multiprocessor decoupled system for the simulation of temporal bone surgery. *Computing and Visualization in Science* 5, 1: 35–43. http://doi.org/10.1007/s00791-002-0085-5

3. Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics* 9, 1: 3–15. http://doi.org/10.1109/TVCG.2003.1175093

4. Yuki Ban, Takashi Kajinami, Takuji Narumi, Tomohiro Tanikawa, and Michitaka Hirose. 2012. Modifying an identified angle of edged shapes using pseudo-haptic effects. *Haptics: Perception, Devices, Mobility, and Communication*: 25–36. Retrieved September 23, 2015 from http://link.springer.com/chapter/10.1007/978-3-642-31401-8_3

5. Yuki Ban, Takashi Kajinami, Takuji Narumi, Tomohiro Tanikawa, and Michitaka Hirose. 2012. Modifying an identified curved surface shape using pseudo-haptic effect. *2012 IEEE Haptics Symposium (HAPTICS)*, IEEE, 211–216. http://doi.org/10.1109/HAPTIC.2012.6183793

6. Yuki Ban, Takuji Narumi, Tomohiro Tanikawa, and Michitaka Hirose. 2012. Modifying an identified position of edged shapes using pseudo-haptic effects. *Proceedings of the 18th ACM symposium on Virtual reality software and technology - VRST '12*, ACM Press, 93–96. http://doi.org/10.1145/2407336.2407353

7. Paul J. Besl and Neil D. McKay. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 2: 239–256. http://doi.org/10.1109/34.121791

8. Anders G. Buch, Dirk Kraft, Joni-Kristian Kamarainen, Henrik G. Petersen, and Norbert Kruger. 2013. Pose estimation using local structure-specific shape and appearance context. *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2080–2087. http://doi.org/10.1109/ICRA.2013.6630856

9. Albert S. Carlin, Hunter G. Hoffman, and Suzanne Weghorst. 1997. Virtual reality and tactile augmentation in the treatment of spider phobia: a case report. *Behaviour Research and Therapy* 35, 2: 153–158. http://doi.org/10.1016/S0005-7967(96)00085-X

10. Kai-Yin Cheng, Rong-Hao Liang, Bing-Yu Chen, Rung-Huei Laing, and Sy-Yen Kuo. 2010. iCon: utilizing everyday objects as additional, auxiliary and instant tabletop controllers. *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, ACM Press, 1155–1164. http://doi.org/10.1145/1753326.1753499

11. Christian Corsten, Ignacio Avellino, Max Möllers, and Jan Borchers. 2013. Instant user interfaces. *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces - ITS '13*, ACM Press, 71–80. http://doi.org/10.1145/2512349.2512799

12. Michael Csongei, Liem Hoang, Ulrich Eck, and Christian Sandor. 2012. ClonAR: Rapid redesign of real-world objects. *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, 277–278. http://doi.org/10.1109/ISMAR.2012.6402572

13. Lisa M. Di Diodato, Richard Mraz, Nicole S. Baker, and Simon J. Graham. 2007. A haptic force feedback device for virtual reality-fMRI experiments. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society* 15, 4: 570–576. http://doi.org/10.1109/TNSRE.2007.906962

14. Markus Funk, Oliver Korn, and Albrecht Schmidt. 2014. An augmented workplace for enabling user-defined tangibles. *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems - CHI EA '14*, ACM Press, 1285–1290. http://doi.org/10.1145/2559206.2581142

15. Sidhant Gupta, Dan Morris, Shwetak N. Patel, and Desney Tan. 2013. AirWave: non-contact haptic feedback using air vortex rings. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp '13*, ACM Press, 419–428. http://doi.org/10.1145/2493432.2493463

16. Steven J. Henderson and Steven Feiner. 2008. Opportunistic controls: leveraging natural affordances as tangible user interfaces for augmented reality. *Proceedings of the 2008 ACM symposium on Virtual reality software and technology - VRST '08*, ACM Press, 211–218. http://doi.org/10.1145/1450579.1450625

17. Valentin Heun, Shunichi Kasahara, and Pattie Maes. 2013. Smarter objects: using AR technology to program physical objects and their interactions. *CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13*, ACM Press, 961–966. http://doi.org/10.1145/2468356.2468528

18. Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. 1994. Passive real-world interface props for neurosurgical visualization. *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*, ACM Press, 452–458. http://doi.org/10.1145/191666.191821

19. Koichi Hirota and Michitaka Hirose. 1995. Providing force feedback in virtual environments. *IEEE Computer Graphics and Applications* 15, 5: 22–30. http://doi.org/10.1109/38.403824

20. Hunter G. Hoffman. 1998. Physically touching virtual objects using tactile augmentation enhances the realism of virtual environments. *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No.98CB36180)*, IEEE Comput. Soc, 59–63. http://doi.org/10.1109/VRAIS.1998.658423

21. Hiroshi Ishii and Brygg Ullmer. 1997. Tangible bits. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, ACM Press, 234–241. http://doi.org/10.1145/258549.258715

22. Hiroo Iwata, Hiroaki Yano, Fumitaka Nakaizumi, and Ryo Kawamura. 2001. Project FEELEX: adding haptic surface to graphics. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, ACM Press, 469–476. http://doi.org/10.1145/383259.383314

23. Benjamin Knoerlein, Gábor Székely, and Matthias Harders. 2007. Visuo-haptic collaborative augmented reality ping-pong. *Proceedings of the international conference on Advances in computer entertainment technology - ACE '07*, ACM Press, 91–94. http://doi.org/10.1145/1255047.1255065

24. Aaron Kotranza and Benjamin Lok. 2008. Virtual Human + Tangible Interface = Mixed Reality Human An Initial Exploration with a Virtual Breast Exam Patient. *2008 IEEE Virtual Reality Conference*, IEEE, 99–106. http://doi.org/10.1109/VR.2008.4480757

25. Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2: 83–97. http://doi.org/10.1002/nav.3800020109

26. Eun Kwon, Gerard J. Kim, and Sangyoon Lee. 2009. Effects of sizes and shapes of props in tangible augmented reality. *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, 201–202. http://doi.org/10.1109/ISMAR.2009.5336463

27. Daniel Leithinger, Sean Follmer, Alex Olwal, et al. 2013. Sublimate: state-changing virtual and physical rendering to augment interaction with shape displays. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*, ACM Press, 1441–1450. http://doi.org/10.1145/2470654.2466191

28. Thomas H. Massie and Kenneth J. Salisbury. 1994. The phantom haptic interface: A device for probing virtual objects. *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, 295–300.

29. Paul Milgram and Fumio Kishino. 1994. A Taxonomy of Mixed Reality Visual Displays. *IEICE*

TRANSACTIONS on Information and Systems E77-D, 1321–1329.

30. Yoichi Ochiai, Kota Kumagai, Takayuki Hoshi, Jun Rekimoto, Satoshi Hasegawa, and Yoshio Hayasaki. 2015. Fairy lights in femtoseconds. *ACM SIGGRAPH 2015 Posters on - SIGGRAPH '15*, ACM Press, 1–1. http://doi.org/10.1145/2787626.2792630

31. Ohan Oda, Levi J. Lister, Sean White, and Steven Feiner. 2008. Developing an augmented reality racing game. 2. Retrieved September 21, 2015 from http://dl.acm.org/citation.cfm?id=1363200.1363203

32. Bernhard Pflesser, Andreas Petersik, Ulf Tiede, Karl Heinz Höhne, and Rudolf Leuwer. 2002. Volume cutting for virtual petrous bone surgery. *Computer aided surgery : official journal of the International Society for Computer Aided Surgery* 7, 2: 74–83. http://doi.org/10.1002/igs.10036

33. Jun Rekimoto. 2014. Traxion: a tactile interaction device with virtual force sensation. *ACM SIGGRAPH 2014 Emerging Technologies on - SIGGRAPH '14*, ACM Press, 1–1. http://doi.org/10.1145/2614066.2614079

34. Radu B. Rusu, Nico Blodow, and Michael Beetz. 2009. Fast Point Feature Histograms (FPFH) for 3D registration. *2009 IEEE International Conference on Robotics and Automation*, IEEE, 3212–3217. http://doi.org/10.1109/ROBOT.2009.5152473

35. Radu B. Rusu. 2010. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz* 24, 4: 345–348. http://doi.org/10.1007/s13218-010-0059-6

36. Satoshi Saga and Ramesh Raskar. 2012. Feel through window. *SIGGRAPH Asia 2012 Emerging Technologies on - SA '12*, ACM Press, 1–3. http://doi.org/10.1145/2407707.2407715

37. Makoto Sato. 2002. SPIDAR and virtual reality. *Proceedings of the 5th Biannual World Automation Congress*, TSI Press, 17–23. http://doi.org/10.1109/WAC.2002.1049515

38. Ruwen Schnabel, Roland Wahl, and Reinhard Klein. 2007. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum* 26, 2: 214–226. http://doi.org/10.1111/j.1467-8659.2007.01016.x

39. Adalberto L. Simeone, Eduardo Velloso, and Hans Gellersen. 2015. Substitutional Reality. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, ACM Press, 3307–3316. http://doi.org/10.1145/2702123.2702389

40. Rajinder Sodhi, Ivan Poupyrev, Matthew Glisson, and Ali Israr. 2013. AIREAL: interactive tactile experiences in free air. *ACM Transactions on Graphics* 32, 4: 134. http://doi.org/10.1145/2461912.2462007

41. Yuriko Suzuki and Minoru Kobayashi. 2005. Air jet driven force feedback in virtual reality. *IEEE Computer Graphics and Applications* 25, 1: 44–47. http://doi.org/10.1109/MCG.2005.1

42. James Vallino and Christopher Brown. 1999. Haptics in augmented reality. *Proceedings IEEE International Conference on Multimedia Computing and Systems*, IEEE Comput. Soc, 195–200. http://doi.org/10.1109/MMCS.1999.779146

43. Peter Weir, Christian Sandor, Matt Swoboda, Ulrich Eck, Gerhard Reitmayr, and Arindam Dey. 2012. BurnAR: Feel the heat. *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, 331–332. http://doi.org/10.1109/ISMAR.2012.6402599

44. Microsoft HoloLens. Retrieved September 21, 2015 from https://www.microsoft.com/microsoft-hololens/en-us

45. Kinect for Xbox One. Retrieved September 21, 2015 from http://www.xbox.com/en-ca/xbox-one/accessories/kinect-for-xbox-one

46. ATOMIC Authoring Tool. Retrieved September 21, 2015 from http://www.sologicolibre.org/projects/atomic/en/

47. buildAR. Retrieved September 21, 2015 from https://buildar.com/start

48. Vuforia Augmented Reality for 3D Mobile Content. Retrieved September 21, 2015 from https://www.qualcomm.com/products/vuforia

49. metaio. Retrieved September 22, 2015 from https://www.metaio.com/

50. dlib C++ Library - Optimization. Retrieved September 21, 2015 from http://dlib.net/optimization.html

51. Meta Augmented Reality. Retrieved September 23, 2015 from https://www.getameta.com/