RealitySketch: Embedding Responsive Graphics and Visualizations in AR through Dynamic Sketching

Ryo Suzuki[1,2,3] **Rubaiat Habib Kazi**[2] **Li-Yi Wei**[2] **Stephen DiVerdi**[2] **Wilmot Li**[2] **Daniel Leithinger**[3]
[1]University of Calgary   [2]Adobe Research   [3]University of Colorado Boulder
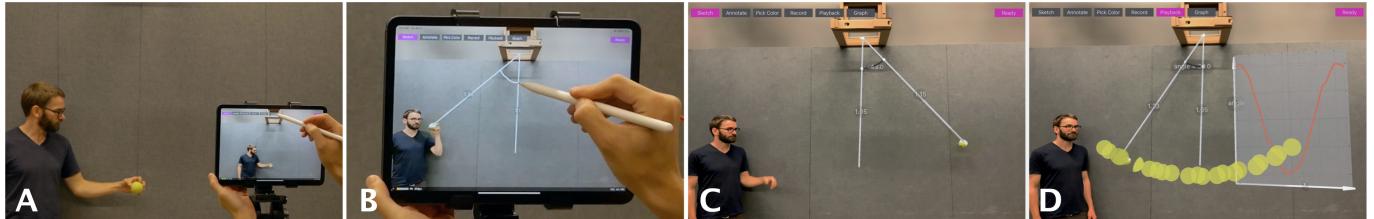ryo.suzuki@ucalgary.ca, {rhabib, lwei, diverdi, wilmotli}@adobe.com, daniel.leithinger@colorado.edu

Figure 1. RealitySketch enables the user to draw and visualize physical phenomena like a pendulum's motion though real-time sketching: A) Select the ball to track, then draw a line and bind it with the physical ball. B) Draw a vertical line and an arc to parameterize the angle between the tracked ball and the center line. C) The sketched elements dynamically respond when the pendulum swings. D) Responsive graphics record and visualize the motion of the pendulum.

ABSTRACT

We present RealitySketch, an augmented reality interface for sketching interactive graphics and visualizations. In recent years, an increasing number of AR sketching tools enable users to draw and embed sketches in the real world. However, with the current tools, sketched contents are inherently *static*, floating in mid air without responding to the real world. This paper introduces a new way to embed *dynamic* and *responsive* graphics in the real world. In RealitySketch, the user draws graphical elements on a mobile AR screen and binds them with physical objects in real-time and improvisational ways, so that the sketched elements dynamically move with the corresponding physical motion. The user can also quickly visualize and analyze real-world phenomena through responsive graph plots or interactive visualizations. This paper contributes to a set of interaction techniques that enable capturing, parameterizing, and visualizing real-world motion without pre-defined programs and configurations. Finally, we demonstrate our tool with several application scenarios, including physics education, sports training, and in-situ tangible interfaces.

CCS Concepts

•**Human-centered computing → Human computer interaction (HCI)**;

Author Keywords

augmented reality; embedded data visualization; real-time authoring; sketching interfaces; tangible interaction;

INTRODUCTION

Over the last decades, interactive and dynamic sketching has been one of the central themes in human-computer interaction (HCI) research [33, 60, 61, 73]. Since Sutherland first demonstrated the power of interactive sketching for computer-aided design [94], HCI researchers have explored ways to sketch *dynamic contents* that can interactively respond to user input [51, 52, 53, 80], thus enabling us to think and communicate through a dynamic visual medium. The applications of such tools are vast, including mathematics and physics education [63, 91], animated art creation [51, 52, 53, 110], and interactive data visualization [9, 11, 64, 70, 100, 109]. Through these research outcomes, we have now developed a rich vocabulary of *dynamic sketching* techniques to fluidly create interactive, animated contents in real-time.

With the advent of augmented and mixed reality interfaces [72], we now have a unique opportunity to expand such practices beyond screen-based interactions towards *reality-based* interactions [43, 45, 104]. In fact, there is an increasing number of tools that provide sketching interfaces for augmented reality, from commercial products like Just a Line [38], Vuforia Chalk [41], and DoodleLens [108] to research projects like SymbiosisSketch [1] and Mobi3DSketch [58]. These tools allow users to sketch digital elements and embed them in the real world. However, a key limitation is that the sketched content is *static* — it does not respond, change, and animate based on user actions or real-world phenomena.

What if, instead, these sketched elements could dynamically respond when the real world changes? For example, imagine a line sketched onto a physical pendulum that moves as the pendulum swings (Figure 1 A-B). This capability would allow us to directly manipulate the sketch through tangible interactions (Figure 1 C) or capture and visualize the pendulum motion to understand the underlying phenomenon (Figure 1 D). Such responsive and embedded sketching would enable a new

way of seeing, understanding, and communicating ideas in much richer ways [79, 101, 102], by bridging the gap between abstract theories and concrete real-world experiences [50].

As a first step toward this goal, we present RealitySketch, an end-user sketching tool to support real-time creation and sharing of embedded interactive graphics and visualization in AR. To create graphics, the user sketches on a phone or tablet screen, which embeds interactive visualizations into a scene. The sketched elements can be bound to physical objects such that they respond dynamically as the real world changes.

Our approach proposes the following four-step workflow: 1) *object tracking*: the user specifies a visual entity (e.g., a physical object, a skeletal joint) to track in the real-world scene; 2) *parameterization*: the user parameterizes tracked entities by drawing lines or arcs that define specific variables of interest; 3) *parameter binding*: the user binds these variables to the graphical properties of sketched elements (e.g., length, angle, etc.) to define their dynamic behavior as the real-world variables change; 4) *visualization*: the user can also interact, analyze, and visualize real-world movements through several visualization effects. We contextualize our approach in the larger design space of dynamic graphics authoring approaches. The main contribution of this paper is a set of interaction techniques that enable these steps without *pre-defined* programs and configurations — rather, the system lets the user perform in *real-time* and *improvisational* ways.

We demonstrate applications of RealitySketch across four domains: augmented physics experiments, mathematical concept explorations, sports and exercise assistants, and creation of improvised tangible UI elements. These examples illustrate how improvisational visualization can enrich interactive experiences and understanding in various usage scenarios. Based on a preliminary evaluation and an expert interview, we discuss the benefits and limitations of the current prototype. Our prototype runs on mobile AR consumer devices that support ARKit (iPhones and iPads), which is important for adoption in an educational context. However, we believe that in the future, our proposed interactions and use cases can also be adapted for head-worn devices.

In summary, this paper makes the following contributions:

1. RealitySketch, a *dynamic sketching* system for real-time authoring of embedded and responsive visualizations in AR, contextualized in the larger design space of dynamic graphics authoring approaches.

2. A set of sketching interaction techniques to parameterize the real-world and bind it with embedded graphics.

3. Application scenarios where the users can create and view in-situ visualizations by capturing physical phenomena, such as for classroom experiments and interactive concept explanations.

RELATED WORK

Sketching Interfaces
In literature of HCI, there is a long history of sketching interface research [3, 24, 33, 34, 62, 73, 94]. Since the space

of sketching interface research is vast, here, we focus on two specific areas of research that are highly relevant.

Sketching Interfaces for VR/AR
In recent years, a growing number of tools support sketching in VR and AR [96]. For example, Just a Line [38], TiltBrush [37], Gravity Sketch [39], and Vuforia Chalk AR [41] are common sketching software available in the market. These tools allow users to sketch objects in which they can walk around [37, 39, 105] or annotate sketches onto a video see-through AR view of the real-world environment [41]. However, due to the challenges of mid-air 3D sketching [2, 5], researchers have leveraged real-world environments and physical objects as contextual guidelines to sketch virtual objects [90]. For example, a physical surface can serve as a geometric constraint for sketching in an immersive environment (e.g., SymbiosisSketch [1], VRSketchIn [16], PintAR [22]), which gives a useful guidance for 3D sketches [2]. Also, previous work utilizes real-world geometry as contextual guidelines for snapping elements (e.g., Mobi3DSketch [58], SnapToReality [77]) or making 3D shapes (e.g., SweepCanvas [67], SketchingWithHands [56]).

However, most of these current sketching tools only support *static* drawings — once sketched, the sketched graphics do not animate or interact with real-world (Figure 2 bottom right). DoodleLens [108] explores the creation of animated drawings with flip-book effects of multiple drawings, but it is not interactive, in the way the sketched elements do not respond to a user's interaction. Video prototyping tools like Pronto [66] also let the user animate embedded sketches, but they only mimic the interaction for prototyping purposes. ProtoAR [76] and 360Proto [75] facilitates the creation of AR and VR experiences with paper sketches, but the sketches themselves are static.

If the sketched graphics become responsive, we can leverage our rich tangible and gestural manipulation capability to interact with the virtual world. For example, in the context of animation authoring, real-world objects provide useful and expressive ways to create dynamic motion by tangible interactions [4, 23, 26]. Human-motion can be also used as real-time inputs for making performance-driven animations [21, 87]. However, these tools often require tedious pre-defined configurations to map between the virtual content and its physical reference (e.g., physical objects, a rig of the body, segmentation of the face), which requires a lot of preparations in advance. It is important to develop more real-time and improvisational ways to specify the behavior because such a practice can preserve a user's natural workflow [54] and let the user focus more on creation and communication [79, 99, 100].

Sketching Dynamic and Responsive Graphics
HCI researchers have also extensively explored sketching interfaces for dynamic and responsive graphics. For example, MathPad2 [63] and PhysInk [91] use hand-drawn sketches to construct an interactive mathematical and physics simulation. In contrast to pre-programmed simulation (e.g., PhET [78, 106], Explorable Explanations [10, 98]), such improvisational sketching enables more flexible applications and encourages natural interaction for human-to-human communica-

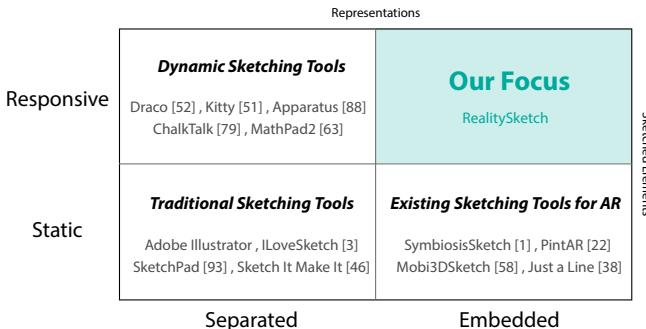|		Representations	
Responsive	**Dynamic Sketching Tools** Draco [52] , Kitty [51] , Apparatus [88] ChalkTalk [79] , MathPad2 [63]	**Our Focus** RealitySketch	Sketched Elements
Static	**Traditional Sketching Tools** Adobe Illustrator , ILoveSketch [3] SketchPad [93] , Sketch It Make It [46]	**Existing Sketching Tools for AR** SymbiosisSketch [1] , PintAR [22] Mobi3DSketch [58] , Just a Line [38]	
	Separated	Embedded	

Figure 2. Design space of dynamic sketching interfaces: The horizontal axis shows whether the representation is embedded (graphics are embedded in the real world) or separated (graphics are isolated from the real world). The vertical axis shows whether sketched elements are dynamic/responsive (sketched elements can dynamically move and respond) or static (sketched elements do not change). Our focus is on embedded and responsive sketching (top right).

tion [64]. This dynamic sketching idea has been implemented for other application domains, such as animation authoring (e.g., Kitty [51], Draco [52], Apparatus [89], Motion Amplifiers [53], and other tools [99, 110, 111]) and dynamic data visualizations (e.g., SketchStory [64], NpakinVis [11], Data Illustrator [70], Transmogrification [9], and Drawing Dynamic Visualizations [100]).

While most of these interfaces are for *screen-based* applications, *augmented reality (AR) based* interfaces have great potential to further enhance interactive experiences and natural communication in the real world. For example, ChalkTalk AR [80, 81] enables sketching dynamic graphics on a see-through display so that the user can present and brainstorm ideas through face-to-face communication. However, these sketched elements do not *respond to* or *interact with* the real-world environments. In other words, the sketched contents are not spatially embedded nor integrated within the real world (Figure 2 top left). In contrast, this paper explore *embedded* responsive sketches that can directly interact with the real world, so that it can provide more immersive experiences for AR and MR (Figure 2 top right).

In the context of tangible user interfaces, there are a few works that explore tangible interactions with sketched elements (e.g., Reactile [95]). Inspired by this line of work, this paper contributes to the discussion and exploration of the new sketching interactions to support a more diverse set of applications.

Augmented Reality and Tangible User Interfaces
Embedded Data and Concept Visualization
Interactive graphics in AR can facilitate communication, education, storytelling, and design exploration. For example, Saquib et al. [87] and Liu et al. [69] demonstrated how interactive and embedded graphics in video presentation can augment storytelling and performance. Similarly, ARMath [47] demonstrates how everyday objects can be used as tangible manipulators to learn basic arithmetics. Data visualization is another promising area, as the user can directly see the data that is associated with physical objects [84]. Willett et al. [107] define such a representation as *embedded visualizations*, in which the visualization is tightly coupled with its physical referent.

Traditionally, such experiences are only available after the dedicated post-production process (e.g., educational videos like [18, 19, 35, 92]), but augmented and mixed reality interfaces promise an *interactive* experience for such concept and data visualizations. For example, in the context of physics education, existing works explore ways to visualize a range of invisible phenomena, such as wind flow (e.g., Urp [97]), optical paths (e.g., HOBIT [20]), terrain simulation (e.g., Illuminating Clay [82]), electromagnetic flow (e.g., [83]), and current flow (e.g., VirtualComponent [57], ConductAR [74]) in the real world. These interfaces can facilitate understanding of a complex idea [17], as viewers can directly observe the hidden phenomenon, instead of imagining it [102]. Beyond education, embedded and interactive visualizations are also promising in application domains like collaborative discussions [15, 48], live storytelling [87], data visualization [13], and sports training [12, 40, 93].

Currently, however, the interactions with these systems are *pre-programmed* for specific applications by their developers, and do not allow non-technical users to adapt them to different contexts or create custom experiences by themselves. With the real-time and improvisational authoring introduced in this paper, we believe end users can easily create, manipulate and share these contents in more engaging and expressive ways, across different application domains.

Object Manipulation through Spatial Tablet Interactions
Spatially aware tablet computers can provide an interface to manipulate virtual and real-world objects. Examples include T(Ether) [59], where users point and drag virtual objects with a spatially tracked tablet computer. ExTouch [49] introduces a similar interaction technique to manipulate robots at a distance. When tapping on a robot shown in an AR video feed, users can drag it to a new target location. By sketching connections between IoT devices, users of Smarter Objects [28] and Reality Editor [27] can program their functionality. Our work extends these ideas of interacting with real-world objects through tablet-based AR. But instead of programming the objects they serve as an input for sketched responsive graphics.

SKETCHING EMBEDDED AND RESPONSIVE GRAPHICS

Definition of Embedded and Responsive Graphics
The goal of this paper is to provide a way to embed dynamic and responsive graphics through dynamic sketching. To better understand the scope, we first define the terminology:

Embedded: graphics are presented and spatially integrated within the real-world

Responsive: graphics change and animate based on the user's interactions

The important aspect of embedded and responsive graphics is that *graphics must interact with the real-world*. Here, the "real-world" means either the environment itself, a physical phenomenon, or a user's tangible and gestural interactions. In the context of AR and tangible user interfaces, there are many non-sketched examples of such embedded and responsive graphics. For example, animated fish in PingPongPlus [44] can interact with the ball by moving towards a point where the physical

A) bouncing balls
C) a swarm of animated fish
E) force of molecular models
G) gravity and force
I) pendulum motion of a swing

B) animated characters
D) wind flow simulation
F) interactive hopscotch
H) brainstorming visualization
J) augmented experiments

Pre-defined Behavior ← – – – – – – – – → User-defined Behavior
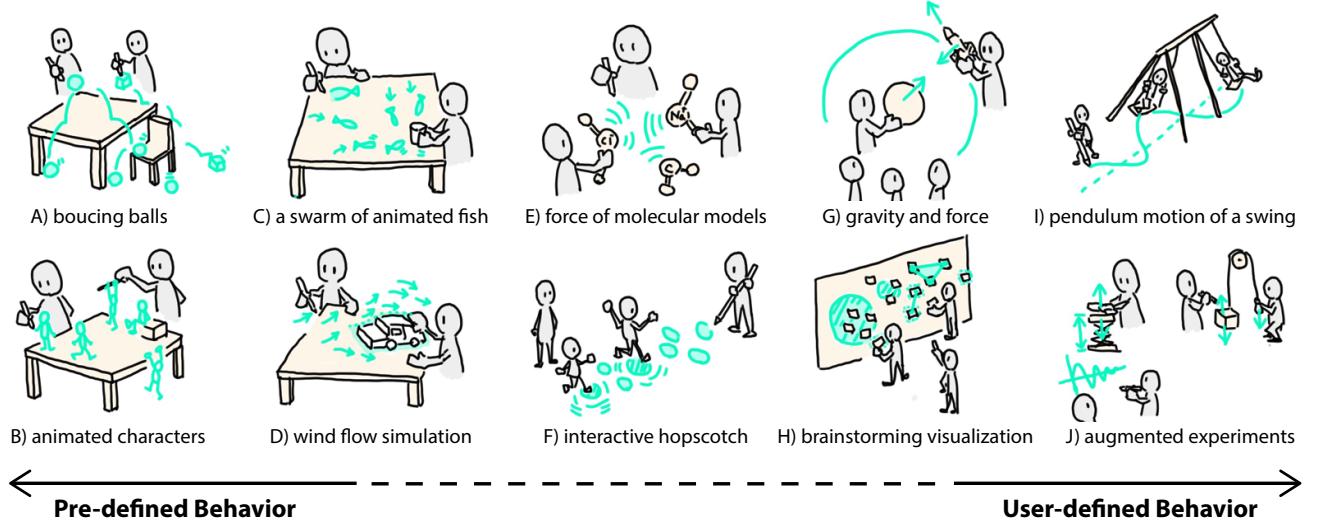
Figure 3. Potential examples of embedded and responsive sketches (green elements represent objects in the virtual world, and black elements represent objects in the real world).

ball hits, or virtual cubes in HoloDesk [29] can interact with physical environments by rolling and bouncing on top of it.

However, to our best of knowledge, few works, if any, investigated *sketching* as a way to create such interactive experiences in real-time. Therefore, this section provides the first exploration and discussion about the interaction spectrum for authoring such embedded and responsive graphics to explore the broader design space. By taking inspirations from the existing works in dynamic sketching of 2D drawing, we discuss and illustrate possible approaches through potential sample applications. We believe this discussion and exploration will help the HCI community to better understand the broader design space and further explore it to fill the gap.

How to Make Embedded Sketches Responsive
In general, crafting responsive and embedded graphics in the real-world can be a continuum between two approaches: predefined behavior and user-defined behavior. Since our focus is on interactive experiences, we do not discuss post-production video editing tools.

Pre-defined Behavior vs User-defined Behavior
Pre-defined behavior refers to a behavior specification given in advance. For example, one can think of a system that specifies all of the sketched elements to follow the law of physics, so that as long as a user draws a virtual element, it automatically falls and bounces on the ground (Figure 3 A). In this case, the behavior of sketched elements is predefined, based on the physics simulation, and the user can only control the shape of the sketches. Similarly, one can imagine a sketched character that starts walking around or interacting with the physical environment (Figure 3 B). In this case, the behavior of the sketched character should also be defined in advance (by programming or design tools), as it is hard to specify such complex behaviors in real-time.

These practices are often utilized in the screen-based sketching interfaces. For example, PhysInk [91] uses a physics engine

and ChalkTalk [80] leverages pre-programmed behavior to animate the sketched elements in real-time.

On the other end of the spectrum, user-defined behavior lets the user decide how the sketched elements move, behave, and animate on the fly. For example, consider an example of visualizing pendulum motion (Figure 1). In this example, the user should be able to specify how and which parameter (e.g., angle) will be visualized (Figure 3 I). In the previous works, Apparatus [89] leverages the user-defined behavior to create interactive diagrams. In this example, the user has full control of how it behaves, based on the user-defined constraints and parameter bindings, which is also known as constraint-based programming [7, 8, 94]. These practices are also utilized to create interactive 2D animation [99], design exploration [65], and dynamic data visualization [70, 100], as it is useful to let the user explicitly specify how it behaves.

We can also think of a design that leverages a combination of both pre-defined and user-defined approaches [111]. For example, consider a sketched school of fish that can automatically swim on a table (Figure 3 C). The basic swimming motion can be specified by pre-defined behavior, but the user could interactively specify the target position so that the fish can continuously chase and gather to a physical object. This can be achieved by by leveraging direct manipulation interfaces that can be paired with the pre-defined program (e.g., Sketch-n-sketch [14], Eddie [88]).

Both approaches have advantages and disadvantages. For example, pre-defined behavior enables highly complex and expressive animation (e.g., Figure 3 A-D), but is targeted at specific applications (e.g., character animation, physics simulation, etc). Beyond these well-known configurations, it is difficult to generalize the range of behaviors and effects. On the other hand, user-defined behaviors can provide building blocks that let the user construct them for different purposes, thus it can be generalizable (e.g., Figure 3 G-J). Also, the user does not need to memorize what behaviors are possible, as it is based on a set of simple interactions. However, creating

complex behaviors with low-level building blocks is challenging, particularly when the interaction is happening in real-time for communication and exploration. Thus, finding the right balance between these two approaches is dependent upon the application and user needs.

Focus of This Paper

Given these considerations, this paper specifically focuses on *user-defined behavior* approach for the input specification and mapping of *embedded* graphics. However, we leverage *pre-defined* visual outputs (e.g., graph plots) so that the user does not need to create these commonly used graphical elements from scratch in real-time. To this end, there are the following research questions:

Q1. Input What is the scope of input values? How can the user visually annotate and define the input value?

Q2. Input-Output Mapping How can the user map the input values to output graphical elements on the fly?

The input, output (graphics), and mapping space can vary across application domains and user needs. We aim to demonstrate interaction techniques for input, output, and mapping that are useful across several application domains (e.g., physics experiments, personal training) where spontaneous and interactive exploration is key to effective performances. In this paper, we answer these research questions by exploring:

A1. Input Parameterizing the real world through sketching interactions. These parameterized values can be later used for the input values of other graphical elements. Bertin [6] introduced a set of visual variables to encode information. In this paper, we focus on continuous spatial variables (e.g., position, distance, angle) as input, as they relate to physical phenomena, tangible interactions, and movements. Other forms of visual variables (e.g., shape, size, type) and gestural interactions are beyond the scope of this paper.

A2. Input-Output Mapping Defining a dynamic parameter value as a variable on-the-fly, so that the variable can be used to map input and output variables.

For visual output, we use pre-defined graphical elements, such as dynamic sketched elements (with the combination of object tracking and parameter binding), responsive visualizations like graph plots, and visual effects like stroboscopic images by recording the motion. We chose these graphical elements due to their effectiveness and versatility to analyze and interact with spatial variables.

REALITYSKETCH: SYSTEM AND DESIGN

Overview

This section introduces RealitySketch, an augmented reality interface to embed responsive graphics and visualizations through real-time dynamic sketching.

RealitySketch supports the followings operations to draw dynamic sketches based on the user-defined specification:

1. Track an object by specifying it on the AR screen.
2. Parameterize the real world with sketching.
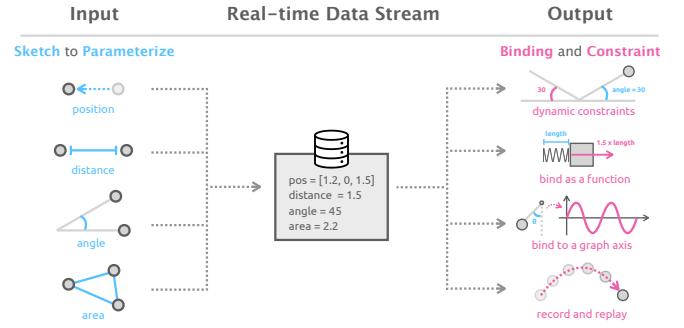**3. Bind** the real-world parameters into existing graphics to make the graphics responsive.

Figure 4. Overview of the interaction workflow: Sketched spatial variables (e.g., position, distance, angle, etc) can be used as dynamic input values. By binding this real-time data stream to existing graphics, the user can create responsive graphics.

4. Visualize the real-world dynamics through responsive graph plots or recording and playing back the motion.

Basic Setup

RealitySketch leverages mobile augmented reality (ARKit) to embed sketches in the real world. The user sketches with a finger or a pen on a touchscreen, where the sketched elements are overlaid onto a camera view of the real world (Figure 5).
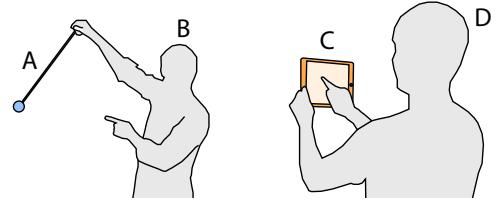

Figure 5. System setup: The user (D) sketches and manipulates on a tablet or phone (C). The user can sketch on a real-time scene or a recorded scene. If necessary, the other user (B) can collaborate to move or interact with physical objects (A).

Rather than 2D sketches that are based on the device screen coordinates, all sketched elements have a 3D geometry and position in real world coordinates, anchored in 3D space. This way, the user can move the device to see from a different perspective and the sketches stay correctly anchored to the real objects.

To enable this functionality, our system leverages ARKit and SceneKit for both surface detection and object placement in the 3D scene. Therefore, the first step is to detect a horizontal or vertical surface, as all interactions (e.g., tracking an object, sketching elements, placing a graph) are based on a reference surface in a 3D scene. Once the surface is detected, the user can start using the system.

In RealitySketch, the user can sketch on both real-time and pre-recorded scenes. For real-time sketching, the user can just start sketching once the surface detection is finished. For recorded scenes, RealitySketch first allows users to capture and record the video with an internal recording feature. The recorded video is embedded with additional meta information at each time frame, such as the current camera position, the current origin point, and the position of detected surface. Based on this recorded information, RealitySketch can reconstruct the 3D scene, which is associated with the recorded camera view of each time frame. Thus, the user can sketch, annotate, or

visualize on top of the RealitySketch's recorded video, while controlling the timeline of the internal video player. Due to these constraints, this recording feature does not work with conventional videos.

Object Tracking

For embedded and responsive graphics, the graphical elements need to be tightly coupled with physical objects and environments. Thus, capturing and tracking an object is vital to make the graphics dynamically change and move.

To specify an object, the user enters the selection mode and then taps an object on the screen (Figure 6A). Once selected, our system highlights the selected object with a white contour line (Figure 6B) and starts tracking the object in the 3D scene (Figure 6C).

Figure 6. Object tracking: Tap an object to be tracked (A), and the system starts tracking the object based on color matching (B-C).

In our current implementation, the system tracks objects based on color tracking implemented with OpenCV. When the user taps an object on the screen, the algorithm gets the HSV value at the x and y position. Then the system captures similar colors at each frame based on a certain upper and lower threshold range. Therefore, the tracking feature best works with distinct, solid colors. Based on this detected mask, the system obtains the largest contour and computes a center position of the tracked object on the screen. The system then converts these 2D coordinates into the 3D world coordinates by ray casting the 2D position onto the detected surface, assuming that the tracked object moves onto the detected horizontal or vertical surface. This color tracking was fast enough for most of our applications (e.g., 20-30 FPS with iPad Pro 11 inch 2018). The user can also use the joint of the human body as additional tracked objects. In this case, we use ARKit 3's built-in human motion tracking feature that gives us the 3D position information of the body joints.
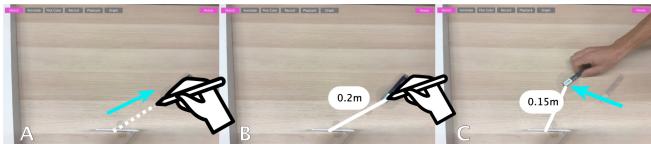

Figure 7. Line segments: Sketch a line on the screen (A). If the line is attached to the tracked object (B), it becomes bound to the object (C).

Parameterization using Line Segments

Next, the user parameterizes the real world to define and capture the dynamic value of interest. In this paper, we specifically focus on parameterization that can be done through simple sketching interactions using *line segments*. Line segments are one of the most basic sketching elements, which are commonly leveraged in the existing interactive sketching systems (e.g., [33]). Line segments are useful as they work alone to parameterize a length between two points, but also can be combined to define different parameters, such as an angle between lines

or an area of a closed geometric shape. They can be also used for annotated lines or guided lines. The user could also draw a simple 3D diagram with these line segments like [32], which could be used as a responsive visual output itself.

Here, we describe how this line sketching interaction can define and capture the dynamic value of interest. First, when entering the sketching mode, the user can start drawing a line segment onto the scene (Figure 7A). All the sketched lines are projected onto a 2D surface within the 3D scene. The system takes the two end-points to create the line segment. This creates a variable that defines the distance between two points on the surface. To create a *dynamic line segment*, the user draws a line whose end point is attached to the selected object (Figure 7B-C). As one end of this *dynamic line segment* is bound to the selected object, if the user moves the object in the real world, the line segment and its parameter (e.g., distance value) will change accordingly. The system visually renders the line segment values using labels.

Figure 8. Angle: The user can draw a path between lines (A), and the system parameterizes it as an angle (B). The angle can be dynamic if the line is bound to the object (C).

RealitySketch employs simple heuristics to determine the nature (e.g., static vs. dynamic, distance vs angle, free move vs constraints, etc) of the line segment. If the start or end point of the line segment is close to an existing tracked object, then the system binds the end point to the tracked object. Thus, for example, if the user draws a line between two tracked objects, then both ends attach to an object. In that case, the line segment captures the distance between those two objects.

The user can also define an area of a closed enclosure or an angle between multiple line segments as a variable. For example, if the user draws a line between the end points of two different lines, then the system binds the three lines together to parameterize the enclosed area of the shape (e.g., a triangle). On the other hand, when the user sketches a stroke between two lines (Figure 8A), the system recognizes it as an angle and shows an arc between the lines (Figure 8B). To determine whether the drawn line is a distance or an angle, the system only considers the start and end points. If both points are attached or close to the existing line segments, the user-drawn line is interpreted as an angle. If the existing line is bound to an object, then the angle is also dynamically changed when the object is moved (Figure 8C). In a similar manner, if the two lines are close to perpendicular, the system creates a perpendicular constraint for two lines (e.g., a tracked line and the ground line in Figure 16).

The values of these parameters (e.g., length, angle, area) are actual physical quantities in the real world, as we can measure the actual length of two points in the 3D scene using ARKit's measuring capability. We use this value for the distance parameter of a sketched line.

The user can also assign a name to an existing parameter e.g., *x, y, angle, length*, etc. This entered name works as a variable that can be later used for input-output mapping. When the user taps the label of the dynamic parameter (Figure 9A), the system shows a popup dialog to let the user define a variable name (Figure 9B). Once the user types the variable name, the system registers it as a variable and shows the name in the label (e.g., *angle* = 40 in Figure 9C).

Figure 9. Naming a variable: By tapping a label of a line segment (A), the user can name a variable (B-C).

Parameter Binding

Make Graphics Responsive based on Parameter Binding

To make the existing line segments responsive, the user can bind variables between two elements. The parameter of a *static line segment* can be bound to another variable. For example, suppose the user has a variable named *angle* for a *dynamic line segment*. When the user taps the label of another angle of the *static line segment* (Figure 10A), the system shows a popup to let the user enter the variable name. If the entered variable name matches an existing name (e.g., *angle* in this case), the angle of the *static line segment* will be dynamically bound to the existing parameter (Figure 10B), so that when the *angle* value changes, the angle of another line is also changed (Figure 10C). This works as a dynamic constraint between multiple line segments.
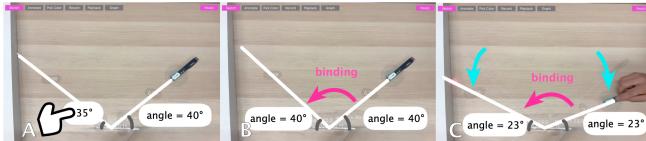

Figure 10. Binding a variable: By using the same variable name for a static line segment (A-B), the variable can be bound to the existing dynamic line segment (C).

Similarly, the user can define a constraint by typing a function of existing variable. For example, consider the user wants to draw the bisector of the angle formed by a *dynamic line segment*. The user can first draw a line and an arc between the line and the base line (Figure 11A). Then, the user taps the label and types *angle*/2 (Figure 11B). Then, the system can recognize the function to dynamically bind the parameter to be half of the existing parameter (Figure 11C). In a similar manner, the user can also change the length or distance of the sketched elements with a function (e.g., the force of a spring in Figure 4). In our current implementation, the system can parse basic arithmetic operations, trigonometric functions, and polynomial expressions.

By default, if the user draws a *dynamic line segment* associated with a tracked object, the other end of the line is fixed to a certain point. However, the user sometimes may want to draw a constant line segment that is attached to an object (e.g., an arrow to represent gravity or force). In such cases, the user
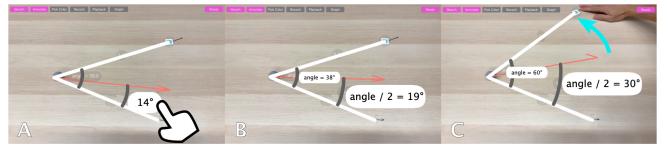

Figure 11. Binding a function: The user can also bind a variable to a function e.g., angle / 2 (A-B), to make a dynamic constraint (C).

can tap annotation option for sketching, so that the distance and direction will be fixed when the tracked object moves (i.e., the endpoint is continuously calculated to maintain the initial relative position from the attached object). This is useful to portray visual hints for hidden forces associated with the object.

Visualizations

Visualize Motion with Responsive Graph Plots

RealitySketch can also make responsive visualization based on graph plotting of a parameter. In the graph placing mode, the user can place a 2D graph and change its size by dragging and dropping onto the surface. By default, the *x*-axis of the graph is time. By binding an existing variable to the graph, it starts visualizing the time series data of the variable.
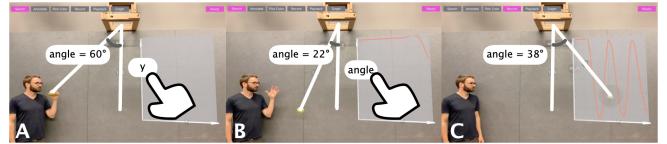

Figure 12. Plotting a graph: By binding an axis label with an existing dynamic variable (A-B), the user can plot a real-time graph of the parameter (C).

To bind the variable, the user can simply tap a label of the graph (Figure 12A), and then, enter the variable the user defined. For example, if the user binds the *angle* variable of the pendulum to the *y*-axis of the chart, the graph will dynamically plot the angle of the pendulum when it starts swinging (Figure 12B-C). By adding multiple variables separated with a comma (e.g., a, b, c), the user can also plot multiple parameters in the same graph. The user can also change the *x*-axis from time to a variable by tapping the *x*-axis and entering a second variable. For example, the user can define an angle and y distance of a point of a circle. By binding x-axis as the angle and y-axis as the perpendicular length, the system can plot the relationship between the angle and corresponding sine value (e.g., Figure 14).

Figure 13. Explanation of the cycloid curve: The user can visualize a point of the rolling circle to see its path is a cycloid curve.

Visualize Motion by Recording and Replaying

Finally, the system also supports recording a motion for analysis. When the user taps the recording mode button, the system records the tracked objects' positions at each frame. The recorded values are stored as an array of 3D positions.

When the user taps the playback button, the system visualizes the trajectory of the motion. To render the time-lapse

Augmented Physics Experiments   **Explorable Concept Explanation**   **Sports Training and Analysis**   **In-situ Tangible User Interfaces**
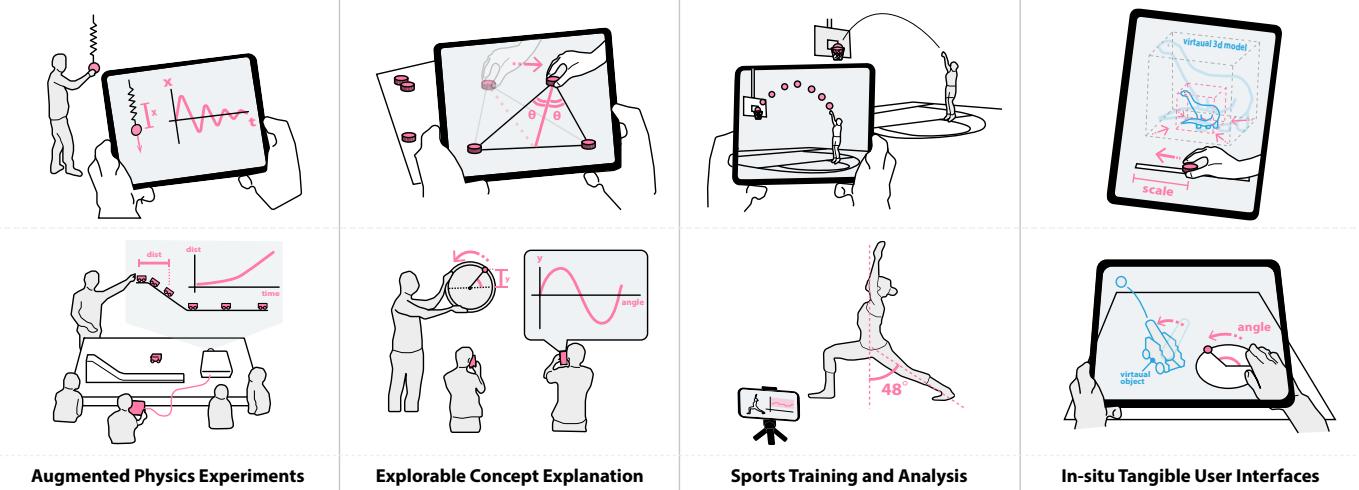
Figure 14. Sketches of proposed application scenarios for RealitySketch.

graphics, the system places 3D spheres of the same tracked color at each recorded 3D position in the world. The user can also change the density of the visualization by changing the minimum distance between each object. For example, if the minimum distance is zero, the system places spheres at all of the recorded positions in the scene (e.g., 200 objects for a 10 second recording at 20 FPS). The default value of the minimum distance is set to 3 cm to better represent the motion and avoid slow performance due to rendering many objects.

APPLICATION SCENARIOS

Augmented Physics Experiments
In science education, a classroom experiment is an integral part of learning physics [103] because the real-world experiment provides students an opportunity to *connect* their knowledge with physical phenomena [30, 71]. RealitySketch can become a powerful tool to support these experiments by leveraging real-time visualization capability. Figure 15 illustrates how our tool can help students understand the pulley system. In this scenario, by tracking the movement of two points (i.e., the point of the person grabs and the point of the load), the students can visualize the traveling distance of each point. In this way, they can see the load distance movement is shorter than the distance the person pulls (Figure 15B-C).
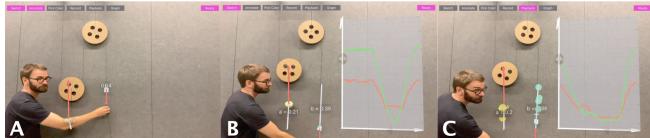

Figure 15. Pulley system experiment: The user measures the distance of a hand movement and a load, and visualizes it with a graph (green: hand movement, orange: the load).

In the same way, our tool can help students visualize the motion of the pendulum experiment (Figure 1 and 12) or behavior of the bouncing spring (Figure 14). In these examples, the user can not only visualize through a graph plot but also add responsive embedded annotations. For example, in Figure 15, the user can add force vector represented with a red line that can dynamically move based on the position. These annotated lines are helpful to let the user see the invisible phenomenon,

such as a path of light reflection in Figure 10 or the resultant force of two pulled strings in Figure 15.

Similarly, recording the trajectory of object motion can be helpful to understand physical phenomena. Figure 16 illustrates an experiment that demonstrates how the law of inertia applies to an object thrown by a moving person. For this experiment, the user first tracks a ball and draws a ground line and vertical line perpendicular to the ground. Then, the user starts tracking the distance of the ball from the ground. By using the recording functionality, the trajectory of the ball is overlaid throughout time. In this way, the student understands even if the person throws a ball in a vertical direction, the ball is continuously moving horizontally. By capturing the height of the ball without lateral movement, the student also understands the vertical trajectory of the ball is the same, by overlaying the two graphs.

Figure 16. Visualizing gravity and inertia force: The system tracks the height of the ball from the ground to visualize how the law of inertia applies to an object launched by a moving person.

Interactive and Explorable Concept Explanation
RealitySketch is also useful to help teachers explain concepts that may be difficult to understand with static graphs, and to let students explore them through tangible interactions. Some examples are shown in Figure 14 (Top: demonstrating an area of a triangle remains the same with horizontal movement; a bisector line of a triangle always intersect at the middle point. Bottom: showing how a sine curve is generated from plotting the angle and perpendicular distance of a rotating point.)

Figure 17 shows an example that explains the relationship between gear ratio and angular motion. By defining and tracking a rotating angle of each gear, the system continuously captures and visualizes the current angles of both gears, which explain the larger gear rotates slowly (e.g., half of the smaller

gear). The system is also useful to explain mathematical concepts. For example, by tracking a point on a rolling circle along a surface, the system can show the trajectory is a cycloid curve (Figure 13). Traditionally, these interactive concept explanations are done through video production [18, 35] or programming [10, 31], but with RealitySketch, it is easier to explain and explore through real-world objects.

Figure 17. Explanation of gear ratios: The teacher explains the angular motion of different sized gears.

For educational applications, we can think of the following three setups of how students and teachers can use our tool:

1. **Classroom presentation:** In this case, a teacher or an assistant sketches and visualizes the motion, which can be shared through a connected large display or projector, so that the students can see and understand.
2. **Collaborative learning:** In this case, students can form a group and interact with their own devices. Since mobile AR is accessible for almost all smartphones, every student should be able to play by themselves, which can lead to an interesting exploration and discoveries.
3. **Remote learning:** In this case, a teacher only records the video of the experiment, then share the recorded video with the students. The student can download and visualize with their own app, so that it provides a fun and interactive experiment to support online and remote learning.

Improvised Visualization for Sports and Exercises

RealitySketch could be also useful to analyze and visualize the motion for sports training and exercises because they often employ the physical movements. For example, sports practices, like a golf shot, baseball pitching, and basketball shooting, would be an interesting example to visualize the trajectory of the ball (Figure 14). Similar to the previous example of showing the trajectory of a ball (Figure 16), it is useful to quickly see the path through stroboscopic effects. In addition to showing the trajectory, the system can also capture and compare multiple attempts to let the user analyze what works and what does not.

Figure 18. Rehabilitation support: The recording feature aids the user's self-training. The user first records the motion, then interactively visualizes the posture by controlling the recorded video.

Also, many sports and exercises may require a proper form and posture. For example, in baseball batting, golf shot, and a tennis swing, a player's form, such as a body angle, can be important, which the tool can help visualize through sketched guided lines. These features could be particularly useful for exercise instructions. For example, in yoga practice, bike riding, or weight lifting training, there are recommended angles

to be followed to maximize the performance benefits. In these cases, the improvisational measurement of the posture can help the user to see and check if correctly done (Figures 18 and 19).

Figure 19. Exercise support: The user measures an angle of body joints for yoga, stretching exercises, and weight lift training, which helps the user check and see if they follow the correct form.

These scenarios can work with co-located or remote instruction where an instructor measures and gives feedback in real-time. It also works as a self-supporting tool in which the user can capture and later analyze it by leveraging the video recording feature. This feature is also useful for rehabilitation assistant. By measuring and visualizing it, the patients can better understand how they should fix it. The body joint tracking supported by ARKit is particularly useful for these scenarios as it provides a more handy way, compared to attaching distinct colored markers on the body.

In-situ Tangible User Interfaces

The parameterized value can be used for many different purposes to enable responsive visual outputs. So far, we have mostly focused on animation of the simple basic geometry (e.g., line segments, arc) or build-in visualization outputs (e.g., graph plot). However, the dynamic parameter value can be also used for other outputs via binding, as we discussed in the previous sections (e.g., pre-defined vs user-defined section).

Figure 20. In-situ creation of a tangible UI slider: The user binds the object distance to the scale of the virtual 3D model, so that the model size changes as the object moves.

One promising application domain of this is to use these dynamic parameter values as an input of changing a property of existing virtual objects. For example, if one can connect a parameter value to a size property of a virtual 3D model, then the size of the model can dynamically change when the value changes. This use case is particularly useful for in-situ creation of tangible controllers (Figure 14). For example, a colored token can become a tangible slider to change the size of the object (Figure 20). The system could bind these values based on a proper naming rule (e.g., *"dino-size"* in Figure 20, *"shark-angle"* in Figure 21 , and *"tree-num"* in Figure 22).

Figure 21. By mapping the defined variable to an angle property of the virtual object, the user can quickly craft an improvised tangible dial to rotate the virtual object.

In addition to sliders, the user can also quickly craft various, improvised controller such as toggle buttons, joysticks, 2D sliders, and dial sliders, by appropriately defining the parameter (e.g., length, position, and angle), and constraints based on expression (e.g., output = (length > 10) to make it binary for the toggle button). Traditionally, making such interactive physical controllers require a dedicated electro-mechanical system (e.g., Arduino) or a well-prepared kit (e.g., Nintendo Labo), but our approach could provide a more handy way to create an in-situ controller for quick use or prototyping, similar to CV-based controllers like ARcadia [55].
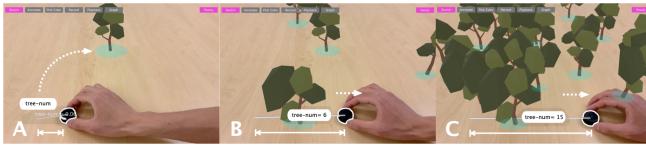

Figure 22. Tracked object parameters can be mapped to a program property through a pre-defined variable name. In this case, the number of virtual trees is controlled by a tangible slider.

PRELIMINARY EVALUATION

Usability Study

Method
We validated our system with two preliminary studies: usability study and expert review. The goal of the first study is to evaluate the usability of our prototype and to identify limitations or opportunities for future improvements. To do so, we recruited six participants (3 male, 3 female, ages 23-33) from our university (3 computer science, 3 engineering). All sessions took place in a research lab, and consisted of two steps. In the first step, the interviewer demonstrates all of the basic functionalities of our prototype system by going through the light reflection example (Figure 6-10). Then, the participants were asked to perform example tasks without the author's help. Due to time constraints, we chose two examples (visualizing the motion of a pendulum and the trajectory of a thrown ball) as evaluation tasks. After the session, we asked the participants to give feedback about the interface and interactions with an online questionnaire form. Sessions lasted approximately 30–45 minutes and participants were compensated USD 10.

Results
All our participants were able to complete the given task without any assistance. In general, participants responded positively to the usability, applications, and unique affordances of RealitySketch. The participants responded that the interactions were intuitive (P1, P2), easy to learn (P5), and enjoyable (P3). "*The interface is simple and highly responsive. Sketching lines and angles, the color-detection, graphing, and playback all felt natural and useful*" (P4). Participants' average rating of the workflow is 5.83 out of 7 (min 5, $\sigma = 0.68$), and the average rating for engagement is 6.83 out of 7 (min 5, $\sigma = 0.37$).

All the participants saw the potential benefits of RealitySketch for classroom teachings, student engagements, enhanced understanding, and in-class work exercises. In particular, participants felt that RealitySketch provides an "*interactive and engaging experience to the students to visually see and involve in creating the abstractions of concepts* " (P3). Participants also stated their desire to use such AR interactions in a variety of applications beyond teaching, including engineering visualization (P1), annotating of worksites (P3), sports analysis (P5), interactive design discussions (P2), and remote consultation for medical applications (P2). "*This system can be used for engineering prototyping, where low-fidelity physical models are annotated by diagrammatic elements*" (P2).

Suggestions for Future Improvements
While the study participants were generally satisfied with RealitySketch, they also gave suggestions for future improvements. For example, the lack of undo and delete operations often hindered their experience (P1, P2, P3, P6). Also, snapping features like [77] were also suggested by the participants to make the drawing easier (P2).

The video recording feature was highly appreciated, but the participants also wanted to analyze pre-existing videos available on YouTube, as there are a lot of interesting resources online (P3, P4). Although our current prototype does not support this feature, machine learning based depth reconstruction of 2D videos [68] could make this possible. The participants also suggested it would be easier to select variables without actually typing them (P3). For example, when tapping the label, it would be better to show the list of pre-defined variables for easier selection. The participants also suggested us to enable the same interactions for spatial augmented reality (e.g., projection mapping), so that multiple users can manipulate the object and share the same experience, which would be more suitable for classroom examples (P3). Finally, they wanted to experience this interface in HMD-based augmented or mixed reality (e.g., Microsoft Hololens), so that the experience can become more immerse.

Expert Review

Method
We also conducted an expert review to gain in-depth feedback from the application scenarios. We recruited six experienced classroom instructors (E1-E6) and sports and yoga instructors (E7-E8) to get expert feedback about how RealitySketch can be integrated into their future practice. E1-E6 have at least 8 years of experience in teaching math- and physics-related classes at the college and high school level, E7 has 2 years of experience in teaching yoga, and E8 has 4 years of experience as a baseball coach. The interviews took place in face-to-face (E1-E4) or remote discussion settings (E5-E8). We first demonstrated the system with a simple example and showed some of the videos to help participants understand the feature. Then, we conducted an in-depth open-ended discussion about our approach and use scenarios. The interview lasted approximately one hour for each expert.

Insights and Feedback
All the participants were very excited about the possibility of this tool. They perceived using a commodity device as a powerful way to help students learn physics (E2, E4, E5). "*I absolutely like the idea of giving them an experiment as homework when you don't need a lab setting, as students don't have that kind of stuff [devices for measuring experiments] at home*" (E2).

Teachers (E3, E5) also mentioned the benefits of using real-world experiments, as compared to computer simulations. E3 saw the possibility of teaching non-intuitive mathematical concepts by enabling students to try them out themselves, referring to an example that connecting the midpoints of an arbitrary quadrilateral forms a parallelogram [7]. "*Say, if you have four colored objects and connect the midpoints of these four lines, then you can interactively see this theorem works by manipulating the objects*" (E3). Such an interactive experience can provide more compelling impressions for students (E1, E3) and allow students to gain intuition for math (E2).

One instructor (E1) shared a suggestion for improvement: "*Currently, the tool can only visualize the experimental data, but it would become more interesting if we can also show and compare the difference between the experimental and theoretical (simulated) data*" (E1). They mentioned that in this way, for example, the tool could show how the ball falls in a different condition, such as on the Earth, Moon, and Mars.

The participants were also excited about the applications for sports and exercise analysis (E7-E8). "*Novice learners often struggle to keep an appropriate posture, so this would be useful for them to check*" (E7). Currently, they often use a mirror to check the posture, but visualizing the body's skeleton through sketching would be useful to analyze the posture (E7). While some felt the sketching interactions might be tedious for everyday use, particularly for expert users like themselves (E7-E8), they also found the sketching interactions interesting as a feature to keep end users engaged (E7-E8).

LIMITATIONS AND FUTURE WORK

More General and Robust Object Tracking

In our current implementation, we use simple color matching for tracking. However, this tracking method becomes unstable if the object does not have a solid color or if objects of similar color exist in the scene. We have also tested with other tracking methods leveraging recent algorithms (e.g., YOLO [85], Faster R-CNN [86], Mask R-CNN [25]). Although these algorithms are more robust under certain circumstances, some of them are slow on a mobile phone when tested (0.1-5 FPS) and require a training period. However, we believe the continuous improvement in real-time computer vision algorithms along with advances in camera hardware (e.g., LiDAR of the iPad Pro 2019) promises to provide more robust, general, and fast tracking methods in the future.

Spatial Interactions for Augmented Storytelling

In mobile AR settings, all interactions are limited to the 2D tablet screen. In the future, we hope to explore how the proposed method can be applied to other configurations (e.g., HMD-based mixed reality or projection mapping) by leveraging spatial and whole-body interactions. Rather than sketching on the mobile device screen, users could for instance sketch directly onto surfaces in the world. Alternative configurations could allow the instructor to directly manipulate and visualize without requiring an assistant. This promises other exciting applications such as interactive storytelling, interaction prototyping, and augmented presentations [87].

Parameterization and Visualization beyond Physical Motion

Currently, our prototype only supports simple constraints, which limits the possible visualization. By leveraging the expressive constraints demonstrated in existing tools (e.g., Sketch it, Make it [46], Shapr3D [42]), we can further expand our idea to more expressive animation and visualizations. Also, our current parameterizations are limited to spatial parameters (e.g., length, angle, area), but there is a broader space of potential real-world parameterizations. Other possibilities include other visual variables (e.g., size, shape, orientation, color, texture, object recognition) [6]. Also, the real-time visualizations can be driven by existing sensors or other commodity devices [36], e.g., plotting voltage by reading a potentiometer value, visualizing a real-world magnetic field by virtually duplicating a magnetic compass, or visualizing acoustic resonance by spatially recording from a mobile phone's microphone. Expanding the scope of input parameters, output graphics (e.g., 3D models), and mapping mechanisms would enable diverse applications to leverage the benefits of embedded and responsive graphics through dynamic sketching. From a design perspective, we would like to further explore the combination of pre-defined procedures to add more complex (procedural) behaviors to user drawn sketches, parameterizations, and physical objects.

Sketching with HMD-based Augmented and Mixed Reality

While we chose mobile AR setup for our prototype primarily because of its wide availability, we also look forward to extending our interactions to head-mounted display (HMD) based setups like Microsoft Hololens to improve immersion. Particularly, in-situ tangible interaction could become a good alternative or complement to current mid-air gestural interactions with HMDs. RealitySketch relies on the mobile device screen for sketching. We expect that this interaction could be adopted to HMD, where the user instead sketches on an existing physical surface (e.g., tabletop) or use the tablet to sketch, then see the sketched elements through the HMD, similar to SymbiosisSketch [1] and PintAR [22].

Deployment to Actual Environments

Our preliminary evaluation is limited in scale and further in-depth evaluation would be required. For future work, we intend to deploy our tool into classroom settings (introductory physics course) for an in-depth and long-term study. This would help us to gain insights about its impact in students' engagement and learning activities.

CONCLUSION

We presented RealitySketch, a tool that allows users to sketch interactive graphics in AR. By interactively binding sketched elements with physical objects, these embedded graphics can dynamically respond when the physical objects move. This paper described the system implementation of RealitySketch and demonstrated the flexibility of the tool through a range of possible applications. We hope this paper opens up new opportunities for embedded and responsive sketching and inspires the HCI community to further explore such interactions to realize the full potential of AR and MR as a dynamic, computational medium.

REFERENCES

[1] Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. 2018. Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 185.

[2] Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George W Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces in VR.. In *CHI*, Vol. 17. 5643–5654.

[3] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 151–160.

[4] Connelly Barnes, David E Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. 2008. Video puppetry: a performative interface for cutout animation. In *ACM Transactions on Graphics (TOG)*, Vol. 27. 124.

[5] Mayra D Barrera Machuca, Paul Asente, Jingwan Lu, Byungmoon Kim, and Wolfgang Stuerzlinger. 2017. Multiplanes: Assisted freehand VR drawing. In *Adjunct Publication of the 30th Annual ACM Symposium on User Interface Software and Technology*. 1–3.

[6] Jacques Bertin. 1983. *Semiology of Graphics*. University of Wisconsin Press.

[7] Alan Borning. 1981. The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 3, 4 (1981), 353–387.

[8] Alan Borning and Robert Duisberg. 1986. Constraint-based tools for building user interfaces. *ACM Transactions on Graphics (TOG)* 5, 4 (1986), 345–374.

[9] John Brosz, Miguel A Nacenta, Richard Pusch, Sheelagh Carpendale, and Christophe Hurter. 2013. Transmogrification: causal manipulation of visualizations. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 97–106.

[10] Nicky Case. 2015. Explorable Explnations. (2015). `https://explorabl.es`

[11] William O Chao, T Munzner, and MVD Panne. 2010. Napkinvis: Rapid pen-centric authoring of improvisational visualizations. In *IEEE Infovis*.

[12] Hua-Tsung Chen, Yu-Zhen He, and Chun-Chieh Hsu. 2018. Computer-assisted yoga training system. *Multimedia Tools and Applications* 77, 18 (2018), 23969–23991.

[13] Zhutian Chen, Yijia Su, Yifang Wang, Qianwen Wang, Huamin Qu, and Yingcai Wu. 2019. MARVisT: Authoring Glyph-based Visualization in Mobile Augmented Reality. *IEEE transactions on visualization and computer graphics* (2019).

[14] Ravi Chugh, Brian Hempel, Mitchell Spradlin, and Jacob Albers. 2016. Programmatic and direct manipulation, together at last. *ACM SIGPLAN Notices* 51, 6 (2016), 341–354.

[15] Pierre Dillenbourg. 2013. Design for classroom orchestration. *Computers & Education* 69 (2013), 485–492.

[16] Tobias Drey, Jan Gugenheimer, Julian Karlbauer, Maximilian Milo, and Enrico Rukzio. 2020. VRSketchIn: Exploring the Design Space of Pen and Tablet Interaction for 3D Sketching in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[17] Matt Dunleavy and Chris Dede. 2014. Augmented reality teaching and learning. In *Handbook of research on educational communications and technology*. Springer, 735–745.

[18] Euphrates. 2008. Mathematical Curves Hidden Around Us. (2008). `https://euphrates.jp/1849115`

[19] Eurosport. 2019. Sports Explainer. (2019). `https://video.eurosport.com/all-sports/video-channel/sports-explainer/`

[20] David Furió, Stéphanie Fleck, Bruno Bousquet, Jean-Paul Guillet, Lionel Canioni, and Martin Hachet. 2017. Hobit: Hybrid optical bench for innovative teaching. In *Proceedings of the 2017 chi conference on human factors in computing systems*. ACM, 949–959.

[21] Emiliano Gambaretto and Charles Piña. 2014. Real-time Animation of Cartoon Character Faces. In *ACM SIGGRAPH 2014 Computer Animation Festival (SIGGRAPH '14)*. 1–1. DOI: `http://dx.doi.org/10.1145/2633956.2658830`

[22] Danilo Gasques, Janet G Johnson, Tommy Sharkey, and Nadir Weibel. 2019. What you sketch is what you get: Quick and easy augmented reality prototyping with pintar. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.

[23] Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobson, Otmar Hilliges, and Olga Sorkine-Hornung. 2016. Rig animation with a tangible and modular input device. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 144.

[24] Mark D Gross and Ellen Yi-Luen Do. 1996. Ambiguous intentions: a paper-like interface for creative design. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*. 183–192.

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[26] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 2012. 3D puppetry: a kinect-based interface for 3D animation.. In *UIST '12*. 423–434.

[27] Valentin Heun, James Hobin, and Pattie Maes. 2013a. Reality editor: programming smarter objects. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. 307–310.

[28] Valentin Heun, Shunichi Kasahara, and Pattie Maes. 2013b. Smarter Objects: Using AR Technology to Program Physical Objects and Their Interactions. In *CHI âĂŹ13 Extended Abstracts on Human Factors in Computing Systems (CHI EA âĂŹ13)*. Association for Computing Machinery, New York, NY, USA, 961âĂŞ966. DOI: `http://dx.doi.org/10.1145/2468356.2468528`

[29] Otmar Hilliges, David Kim, Shahram Izadi, Malte Weiss, and Andrew Wilson. 2012. HoloDesk: direct 3d interactions with a situated see-through display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2421–2430.

[30] Derek Hodson. 1996. Laboratory work as scientific method: Three decades of confusion and distortion. *Journal of Curriculum studies* 28, 2 (1996), 115–135.

[31] Joshua Horowitz. 2015. Gallery of Concept Visualization. (2015). `https://conceptviz.github.io/`

[32] Takeo Igarashi and John F Hughes. 2007. A suggestive interface for 3D drawing. In *ACM SIGGRAPH 2007 courses*. 20–es.

[33] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. 2007. Interactive beautification: a technique for rapid geometric design. In *ACM SIGGRAPH 2007 courses*. 18–es.

[34] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 2006. Teddy: a sketching interface for 3D freeform design. In *ACM SIGGRAPH 2006 Courses*. 11–es.

[35] Byju's Inc. 2019a. Byju's The Learning App. (2019). `https://byjus.com`

[36] Google Inc. 2016a. Google Science Journal. (2016). `https://sciencejournal.withgoogle.com`

[37] Google Inc. 2016b. TiltBrush. (2016). `https://www.tiltbrush.com/`

[38] Google Inc. 2018. Just a Line. (2018). `https://justaline.withgoogle.com/`

[39] Gravity Sketch Inc. 2017a. Gravity Sketch. (2017). `https://www.gravitysketch.com/`

[40] NEX Team Inc. 2019b. HomeCourt. (2019). `https://www.homecourt.ai/`

[41] PTC Inc. 2017b. Vuforia Chalk AR. (2017). `https://chalk.vuforia.com/`

[42] Shapr3D Inc. Shapr3D Constraint. (????). `https://support.shapr3d.com/hc/en-us/articles/115001416214-Constraints`

[43] Hiroshi Ishii and Brygg Ullmer. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. 234–241.

[44] Hiroshi Ishii, Craig Wisneski, Julian Orbanes, Ben Chun, and Joe Paradiso. 1999. PingPongPlus: design of an athletic-tangible interface for computer-supported cooperative play. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 394–401.

[45] Robert JK Jacob, Audrey Girouard, Leanne M Hirshfield, Michael S Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. 2008. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 201–210.

[46] Gabe Johnson, Mark Gross, Ellen Yi-Luen Do, and Jason Hong. 2012. Sketch it, make it: sketching precise drawings for laser cutting. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. ACM, 1079–1082.

[47] Seokbin Kang, Ekta Shokeen, Virginia L Byrne, Leyla Norooz, Elizabeth Bonsignore, Caro Williams-Pierce, and Jon E Froehlich. 2020. ARMath: Augmenting Everyday Life with Math Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–15.

[48] Shunichi Kasahara, Valentin Heun, Austin S Lee, and Hiroshi Ishii. 2012. Second surface: multi-user spatial collaboration system based on augmented reality. In *SIGGRAPH Asia 2012 Emerging Technologies*. 1–4.

[49] Shunichi Kasahara, Ryuma Niiyama, Valentin Heun, and Hiroshi Ishii. 2013. exTouch: spatially-aware embodied manipulation of actuated objects mediated by augmented reality. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*. 223–228.

[50] Alan Kay. 2005. Squeak etoys, children & learning. *online article* 2006 (2005).

[51] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014a. Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 395–405.

[52] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014b. Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 351–360.

[53] Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Motion amplifiers: sketching dynamic illustrations using the principles of 2D animation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4599–4609.

[54] Rubaiat Habib Kazi, Takeo Igarashi, Shengdong Zhao, and Richard Davis. 2012. Vignette: interactive texture design and manipulation with freeform gestures for pen-and-ink illustration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1727–1736.

[55] Annie Kelly, R Benjamin Shapiro, Jonathan de Halleux, and Thomas Ball. 2018. ARcadia: A rapid prototyping platform for real-time tangible interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–8.

[56] Yongkwan Kim and Seok-Hyung Bae. 2016. SketchingWithHands: 3D sketching handheld products with first-person hand posture. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 797–808.

[57] Yoonji Kim, Youngkyung Choi, Hyein Lee, Geehyuk Lee, and Andrea Bianchi. 2019. VirtualComponent: a Mixed-Reality Tool for Designing and Tuning Breadboarded Circuits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 177.

[58] Kin Chung Kwan and Hongbo Fu. 2019. Mobi3DSketch: 3D Sketching in Mobile AR. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 176.

[59] David Lakatos, Matthew Blackshaw, Alex Olwal, Zachary Barryte, Ken Perlin, and Hiroshi Ishii. 2014. T (ether): spatially-aware handhelds, gestures and proprioception for multi-user 3D modeling and animation. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*. 90–93.

[60] James A Landay. 1996. SILK: sketching interfaces like krazy. In *Conference companion on Human factors in computing systems*. 398–399.

[61] James A Landay and Brad A Myers. 1995. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 43–50.

[62] James A Landay and Brad A Myers. 2001. Sketching interfaces: Toward more human interface design. *Computer* 34, 3 (2001), 56–64.

[63] Joseph J LaViola Jr and Robert C Zeleznik. 2004. MathPad 2: a system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 432–440.

[64] Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. 2013. SketchStory: Telling more engaging stories with data through freeform sketching. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2416–2425.

[65] Joon Hyub Lee, Hyung-Gi Ham, and Seok-Hyung Bae. 2020. 3D Sketching for Multi-Pose Products. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–8.

[66] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[67] Yuwei Li, Xi Luo, Youyi Zheng, Pengfei Xu, and Hongbo Fu. 2017. SweepCanvas: Sketch-based 3D prototyping on an RGB-D image. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 387–399.

[68] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. 2019. Learning the depths of moving people by watching frozen people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4521–4530.

[69] Jingyuan Liu, Hongbo Fu, and Chiew-Lan Tai. 2020. PoseTween: Pose-driven Tween Animation. In *UIST '20*.

[70] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. 2018. Data Illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 123.

[71] William F McComas, Michael P Clough, and Hiya Almazroa. 1998. The role and character of the nature of science in science education. In *The nature of science in science education*. Springer, 3–39.

[72] Paul Milgram and Fumio Kishino. 1994. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems* 77, 12 (1994), 1321–1329.

[73] Elizabeth D Mynatt, Takeo Igarashi, W Keith Edwards, and Anthony LaMarca. 1999. Flatland: new dimensions in office whiteboards. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 346–353.

[74] Koya Narumi, Steve Hodges, and Yoshihiro Kawahara. 2015. ConductAR: an augmented reality based tool for iterative design of conductive ink circuits. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 791–800.

[75] Michael Nebeling and Katy Madier. 2019. 360proto: Making Interactive Virtual Reality & Augmented Reality Prototypes from Paper. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[76] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. Protoar: Rapid physical-digital prototyping of mobile augmented reality applications. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[77] Benjamin Nuernberger, Eyal Ofek, Hrvoje Benko, and Andrew D Wilson. 2016. Snaptoreality: Aligning augmented reality to the real world. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1233–1244.

[78] Katherine Perkins, Wendy Adams, Michael Dubson, Noah Finkelstein, Sam Reid, Carl Wieman, and Ron LeMaster. 2006. PhET: Interactive simulations for teaching and learning physics. *The physics teacher* 44, 1 (2006), 18–23.

[79] Ken Perlin. 2016. Future Reality: How emerging technologies will change language itself. *IEEE computer graphics and applications* 36, 3 (2016), 84–89.

[80] Ken Perlin, Zhenyi He, and Karl Rosenberg. 2018a. Chalktalk: A Visualization and Communication Language–As a Tool in the Domain of Computer Science Education. *arXiv preprint arXiv:1809.07166* (2018).

[81] Ken Perlin, Zhenyi He, and Fengyuan Zhu. 2018b. Chalktalk VR/AR. *International SERIES on Information Systems and Management in Creative eMedia (CreMedia)* 2017/2 (2018), 30–31.

[82] Ben Piper, Carlo Ratti, and Hiroshi Ishii. 2002. Illuminating clay: a 3-D tangible interface for landscape analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 355–362.

[83] Iulian Radu and Bertrand Schneider. 2019. What Can We Learn from Augmented Reality (AR)?. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 544.

[84] Ramesh Raskar, Paul Beardsley, Jeroen van Baar, Yao Wang, Paul Dietz, Johnny Lee, Darren Leigh, and Thomas Willwacher. 2004. RFIG lamps: interacting with a self-describing world via photosensing wireless tags and projectors. In *ACM SIGGRAPH 2004 Papers*. 406–415.

[85] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[86] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[87] Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. 2019. Interactive Body-Driven Graphics for Augmented Video Performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 622.

[88] John Sarracino, Odaris Barrios-Arciga, Jasmine Zhu, Noah Marcus, Sorin Lerner, and Ben Wiedermann. 2017. User-Guided Synthesis of Interactive Diagrams. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 195–207.

[89] Toby Schachman. 2015. Apparatus: a hybrid graphics editor/programming environment for creating interactive diagrams. *Strange Loop* (2015).

[90] Steven Schkolne, Michael Pruett, and Peter Schröder. 2001. Surface drawing: creating organic 3D shapes with the hand and tangible tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 261–268.

[91] Jeremy Scott and Randall Davis. 2013. Physink: sketching physical behavior. In *Proceedings of the adjunct publication of the 26th annual ACM symposium on User interface software and technology*. ACM, 9–10.

[92] PBS: Public Broadcasting Service. 2019. Nova. (2019). `https://www.pbs.org/wgbh/nova/`

[93] L Sousa, R Alves, and JMF Rodrigues. 2016. Augmented reality system to assist inexperienced pool players. *Computational Visual Media* 2, 2 (2016), 183–193.

[94] Ivan E Sutherland. 1964. Sketchpad a man-machine graphical communication system. *Simulation* 2, 5 (1964), R–3.

[95] Ryo Suzuki, Jun Kato, Mark D Gross, and Tom Yeh. 2018. Reactile: Programming Swarm User Interfaces through Direct Physical Manipulation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 199.

[96] Balasaravanan Thoravi Kumaravel, Cuong Nguyen, Stephen DiVerdi, and Björn Hartmann. 2019. TutoriVR: A Video-Based Tutorial System for Design Applications in Virtual Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[97] John Underkoffler and Hiroshi Ishii. 1999. Urp: a luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 386–393.

[98] Bret Victor. 2011. Explorable explanations. *Bret Victor* 10 (2011).

[99] Bret Victor. 2012. Stop drawing dead fish. In *ACM SIGGRAPH*.

[100] Bret Victor. 2013. Drawing dynamic visualizations. *URL: worrydream. com* (2013).

[101] Bret Victor. 2014a. Humane representation of thought: a trail map for the 21st century. In *Proceedings of the companion publication of the 2014 ACM SIGPLAN conference on Systems, Programming, and Applications: Software for Humanity*. 5–5.

[102] Bret Victor. 2014b. Seeing spaces. In *Talk at EG conference*.

[103] Jerry Wellington. 1998. *Practical work in school science: Which way now?* Routledge.

[104] Pierre Wellner. 1993. Interacting with paper on the DigitalDesk. *Commun. ACM* 36, 7 (1993), 87–96.

[105] Gerold Wesche and Hans-Peter Seidel. 2001. FreeDrawer: a free-form sketching system on the responsive workbench. In *Proceedings of the ACM symposium on Virtual reality software and technology*. 167–174.

[106] Carl E Wieman, Wendy K Adams, and Katherine K Perkins. 2008. PhET: Simulations that enhance learning. *Science* 322, 5902 (2008), 682–683.

[107] Wesley Willett, Yvonne Jansen, and Pierre Dragicevic. 2016. Embedded data representations. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 461–470.

[108] Aidan Wolf. 2019. DoodleLens. (2019). `https://www.doodlelens.app/`

[109] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. 2018. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 223, 13 pages. `DOI: http://dx.doi.org/10.1145/3173574.3173797`

[110] Jun Xing, Rubaiat Habib Kazi, Tovi Grossman, Li-Yi Wei, Jos Stam, and George Fitzmaurice. 2016. Energy-brushes: Interactive tools for illustrating stylized elemental dynamics. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 755–766.

[111] Bo Zhu, Michiaki Iwata, Ryo Haraguchi, Takashi Ashihara, Nobuyuki Umetani, Takeo Igarashi, and Kazuo Nakazawa. 2011. Sketch-based dynamic illustration of fluid systems. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–8.