ArticuLev: An Integrated Self-Assembly Pipeline for Articulated Multi-Bead Levitation Primitives

Andreas Rene Fender
ETH Zurich

Diego Martinez Plasencia
University College London

Sriram Subramanian
University College London

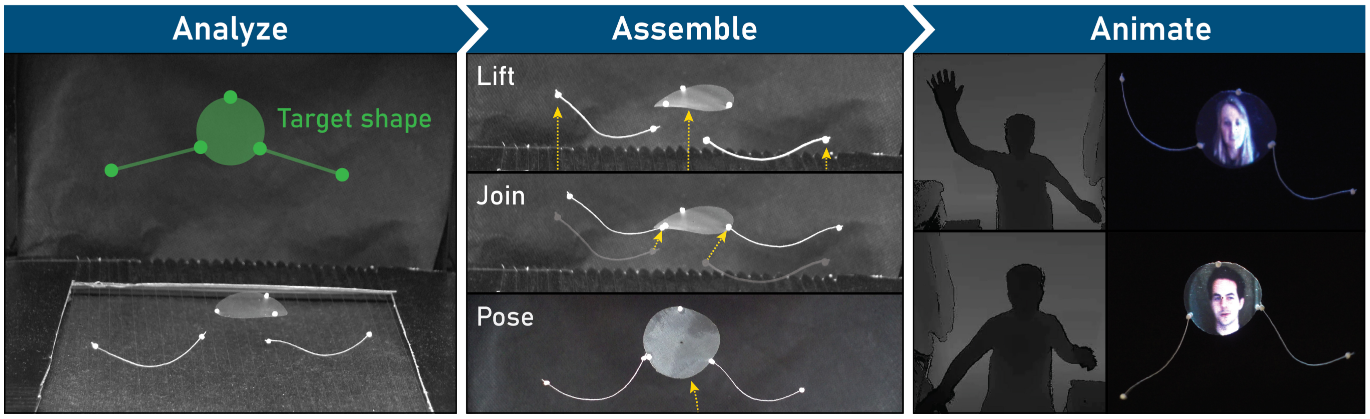

Figure 1. ArticuLev provides an integrated pipeline for the identification, assembly and mid-air placement of shape primitives. The developer specifies the target structure of primitives required ("Target shape", green). The pipeline automatically matches existing primitives to the intended shape (*Analyze*). After lifting the primitives, ArticuLev joins them and manipulates the shape in mid-air to match the target pose (*Assemble*). The levitated shapes can be programmed and manipulated in real-time (*Animate*) and easily combined with input/output devices (e.g., *Microsoft Kinect* and projectors).

ABSTRACT

Acoustic levitation is gaining popularity as an approach to create physicalized mid-air content by levitating different types of levitation primitives. Such primitives can be independent particles or particles that are physically connected via threads or pieces of cloth to form shapes in mid-air. However, initialization (i.e., placement of such primitives in their mid-air target locations) currently relies on either manual placement or specialized ad-hoc implementations, which limits their practical usage. We present ArticuLev, an integrated pipeline that deals with the identification, assembly and mid-air placement of levitated shape primitives. We designed ArticuLev with the physical properties of commonly used levitation primitives in mind. It enables experiences that seamlessly combine different primitives into meaningful structures (including fully articulated animated shapes) and supports various levitation display approaches (e.g., particles moving at high speed). In this paper, we describe our pipeline and demonstrate it with heterogeneous combinations of levitation primitives.

CCS Concepts

•Human-centered computing → User interface toolkits;

CHI '21 May 8–13, 2021 in Yokohama, Japan

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-8096-6/21/05...$15.00

DOI: 10.1145/3411764.3445342

Author Keywords

Acoustic Levitation, Mid-air UIs, Pipelines, Toolkits.

INTRODUCTION

Recent advancements in acoustic levitation enable levitation and movement of multiple lightweight objects like polystyrene beads [15, 16], with increased adoption for new types of mid-air displays [29]. To display content in mid-air, some approaches use sparse sets of beads [23, 24], threads [25, 15] or even organza cloth, which can also serve as levitating surface for projection mapping [19]. Most recently, continuous, fully volumetric content has also been demonstrated using single [9] or multiple [12] quickly moving beads to exploit the persistence of vision (PoV) effect. Inspired by the potential of acoustic levitation, the HCI community has started to explore interaction issues around levitation interfaces [7, 1], geometry and content creation [19, 6] or even simulators [26].

Unlike traditional displays which are in full control of the delivery of content (e.g., by illuminating pixels), levitation-based displays require external physical props (i.e., the beads, threads or cloth) to be placed within their working volume. Displaying content in mid-air through levitation is thus hindered, because such *levitation primitives* need to be identified, picked up by the system (or a human operator) and physically transported to the target location via moving acoustic traps before any content can be presented. Such detection and assembly steps were mostly neglected and never formalized in previous research and were instead implemented as specialized ad-hoc solutions. More specifically, there are no approaches to automatically detect levitation primitives and assemble them in mid-air to allow for arbitrary combinations of primitives or fully articulated animations.

We present ArticuLev, a system for self-supported articulated shapes which deals with the necessary steps to initialize mid-air content for interactive levitation applications. The system supports all the levitation primitives used to date in mid-air levitated displays (i.e., independent beads, threads and cloth) and existing content presentation approaches (e.g., animated shapes, PoV and projection mapping). We achieve this by considering and exploiting the physical properties of the materials used as primitives (i.e. light scattering, topology), as well as the capabilities of the levitator system (e.g., traps and interactions while merging), resulting in an integrated pipeline. In ArticuLev, a *developer* (i.e., in the context of this paper, a person or group that wants to build acoustic levitation experiences) simply defines target shapes and implements the logic in Unity3D to animate them, without dealing with the underlying detection, assembly or acoustic traps. When placing matching levitation primitives into the levitator, our system automatically analyses and assembles them, to then run the developer's application. Figure 1 shows an example of the detection and assembly process of a shape consisting of three primitives (i.e., two threads and a piece of cloth), for an embodied teleconference application. We contribute the following:

- A detection approach that not only detects bead locations, but also how they are connected, e.g., via threads and cloth, matching them with developer-defined target shapes.
- An assembly method to create animated articulated levitated shapes made of heterogeneous levitation primitives (by joining primitives via trap merging).
- A set of examples to demonstrate the creation of heterogeneous levitated interfaces that combine different types of primitives (beads, threads, cloth) and display approaches (e.g., PoV and projection mapping).

In this paper, we describe the ArticuLev pipeline. We explain the stages to successfully identify beads and the materials that connect them as they are placed in the levitator, matching them with easily configurable target shapes. After that, we elaborate on the assembly steps which involve picking up the identified primitives and joining them into target shapes. We then provide a technical evaluation to test ArticuLev's ability to detect and assemble various levitated primitives. Lastly, we demonstrate the wide applicability of our pipeline through various interactive prototypes, mixing different types of primitives and/or display approaches, as well as existing interactive concepts like 2D/3D input or rigid body simulations.

RELATED WORK

Levitation has long been a research interest in physics [3]. Several of these approaches have been adopted to create mid-air levitated displays, trapping physical display elements with either magnetic fields [10], optical and electromagnetic traps [30, 2] and, most prominently within HCI, acoustic fields.

Several setups enable exploitation of acoustic levitation fields, such as transducer-reflector setups creating standing waves [33] or metamaterials [18, 22], but their limited manipulation capabilities hardly make them suitable for display approaches. Phased Arrays of Transducers (PATs) provide electronic control of the phase and amplitude of densely packed sets of transducers (e.g., 16×16 transducers). Such transducer boards are the most prominent approach for acoustic levitation displays (or other formats such as gloves [13]). Top-bottom PATs made displays such as JOLED possible [29], which not only levitated coated particles, but also controlled their orientation. Controlled phase patterns allowed constrained control of single particles and particles attached to threads [25], or clusters of particles moved as a group [23].

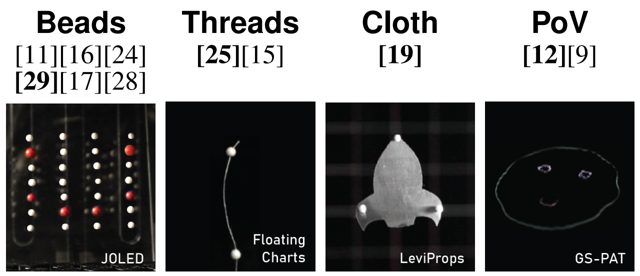

Figure 2. Types of primitives used to display content in levitated mid-air displays from previous literature (references in bold are the sources of the example pictures). *JOLED* uses beads with two colors to generate mid-air pixels. *Floating Charts* uses threads for data visualization. *LeviProps* levitates cloth cutouts that represent specific objects. *GS-PAT* uses high-speed moving beads to create PoV volumetric content.

Holographic Acoustic Elements (HAEs) [16] formalized the definition of acoustic levitation holograms in terms of *signature* and *focusing patterns*, enabling free 3D control of single particles and modular display systems such as LeviPath [24], with several particles, each in an independent, small, top-bottom PAT. HAEs were later extended to allow free 3D control of several particles [15]. This enabled 3D display shapes made of particles (e.g. icoshaedrons) and display systems combining such shapes with external (non-levitated) cloth props [6]. *LeviProps* [19] extended these by levitating cloth cutouts, providing an algorithm to optimize the location of beads on the cloth and enabling levitated 3D projection screens. Fast computation of acoustic levitation fields have allowed the creation of continuous levitated 3D content by using fast moving beads (one bead [9, 8] or multiple beads [12]), while simultaneously creating mid-air haptics and audio.

In summary, previous work enabled new independent functionalities (e.g. PoV content [12], optimal prop design [19]). However, none of them formalized the initial placement of the beads and props in mid-air (i.e. a preliminary step required before their functionality can be leveraged) and none supported heterogeneous combinations of levitation primitives. Besides, no prior system has attempted to generate articulated levitated shapes, i.e., shapes which consists of multiple parts that can be transformed hierarchically. With this, not only does our system generalize previous systems that involve connected beads, but it also enables new forms of interactive levitated displays. The types of levitation primitives from previous work (see Figure 2) and the commonly used top-bottom PAT arrangement (see transducer boards in Figure 3) frame the scope of our design space.

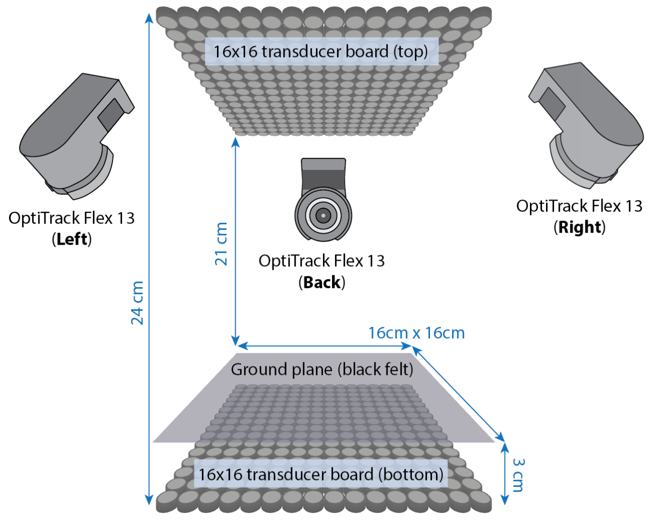
16x16 transducer board (top)
OptiTrack Flex 13 (Left)
OptiTrack Flex 13 (Right)
OptiTrack Flex 13 (Back)
21 cm
24 cm
16cm x 16cm
Ground plane (black felt)
16x16 transducer board (bottom)
3 cm

Figure 3. Setup of our pipeline implementation. We use two transducer boards with 16×16 transducers each (i.e., 512 total) in a top-bottom arrangement. The ground plane consists of acoustically transparent black felt. For detection, we use three *OptiTrack Flex 13* cameras.

ARTICULEV: OVERVIEW AND REQUIREMENTS

ArticuLev is a self-contained detection and levitation system supporting the steps required to initialize and run levitation-based mid-air experiences. Developers can create levitation scenes by defining various animated *target shapes*. The developer defines the initial state of those target shapes (e.g., the initial posture as in Figure 1, left) and programs the application logic starting from this initial state (in C# within Unity3D). Given this input and logic, ArticuLev deals with the required initialization and execution in three stages. In the *Analyze* stage, we use the cameras to *detect* the primitives laid into the levitator and *match* those with the defined target shapes. After that, the *Assemble* stage arranges the levitation primitives into their initial shape and pose. In the final *Animate* stage the developer's logic is executed.

All of this functionality of the pipeline relies on the exploitation of implicit properties of levitated display systems. This section provides background information on such properties (i.e., supported materials, hardware and software), as they influence both the design, operation and applicability of our approach. Afterwards, we describe the developer's input and the pipeline stages in the subsequent sections.

Primitive materials

We use the most common materials from related literature for our primitives (see Figure 2). More specifically, we use white polystyrene beads as *bead primitives*, white cotton thread for *thread primitives*, and SuperOrganza for *cloth primitives*. Beads and threads provide an almost Lambertian surface, with diffuse reflections that are easily detectable from any viewing angle. In contrast, reflections from SuperOrganza contain much contributions from specular reflections, making it view-dependent. Those material properties have a direct impact on the design of our detection algorithm which we will describe in *Detecting beads and connections*.

Hardware and software

ArticuLev is not bound to specific hardware and software components, but there are requirements that need to be met for our pipeline to work. In this section, we describe those requirements as well as the components that we chose for our specific implementation.

Levitator

We use a top-bottom levitation setup (see Figure 3), which is the most commonly used setup in current levitation experiences [16, 19, 12, 9, 1]. The setup consists of two 16×16 arrays of 40KHz ultrasound transducers, created as an extension of the open source platform Ultraino [14]. We use similar voltage (20V) and transducers as CE certified products that are safe for public use (e.g., *Ultraleap STRATOS* [34]). This top-bottom setup provides a working volume of approximately 10cm×10cm×18cm at the center of the levitator [19] and produces acoustic traps that are roughly 25mm wide horizontally [15, 9]. This determines the minimum distances at which beads (individual or as part of primitive) can be laid, as to allow independent picking and manipulation (to avoid unintended trap merging). We also use a layer of black felt (~0.2mm thickness, acoustically transparent and IR absorbent) placed 3cm above the bottom array. This *ground plane* acts as a support structure to lay the primitives and as a dark background for detection.

For generating acoustic traps, we implemented a *Naive* multi-point levitation algorithm [15, 12] multi-threaded on the CPU in C#, computing up to 3K solutions per second (we use a laptop with six i7 cores / twelve logical processors at 2.2GHz). This update rate provides basic support for PoV content. ArticuLev is agnostic to the underlying algorithm used, i.e., other algorithms can be used to improve stability and update rate (e.g., IBP [15] or GS-PAT [12], which is GPU-based and supports over 10K solutions per second).

Camera setup

We used three infrared-based cameras (*OptiTrack Flex 13* by *NaturalPoint* [21], 1280×1024 pixels each), as a widely available tracking solution. These cameras are robust in terms of varying visible light conditions and have the ability to dynamically adjust IR lighting intensity and exposure time, required to adapt captured images to the optical properties of the potential primitives to be detected. The camera *intrinsics* and distortion coefficients are provided by the manufacturer. Therefore, we only need to compute the camera *extrinsics* by estimating one homography [4] per camera, for which we use the four corners of our *ground plane*. Other IR or RGB cameras can be used, assuming equivalent capabilities are provided.

Software frameworks and libraries

The majority of the system is implemented in the *Unity3D Game Engine*, to easily exploit its 3D programming capabilities and to enable a wide variety of levitation applications. The overall dataflow implementation is based on the *Velt* framework [5] (Unity plugin), which facilitates multi-threaded camera frame and 3D input processing. Therefore, those and other orthogonal functionalities (e.g., projection mapping) are not integral parts of the ArticuLev pipeline. Lastly, we use *alglib* [27] to solve linear equations within various pipeline steps.

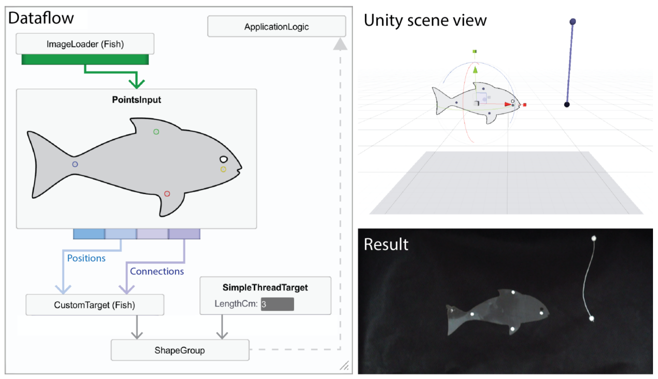

Figure 4. Creating an example ArticuLev application with an animated fish and a line and sinker. Left: *Velt* nodes are used to describe the target shapes in the application, specifying their bead positions, connections and cloth outlines. The `ShapeGroup` node gathers the two target shapes and the `ApplicationLogic` handles animations or interactive experiences. Top-right: Unity scene view, allowing placement of target shapes. Bottom-right: Resulting application.

DEVELOPER'S INPUT

Development with ArticuLev requires basic Unity skills. Our target group of developers includes programmers, HCI researchers and digital artists. Figure 4 illustrates the developer's creation process, defined visually using *Velt* nodes (boxes in Figure 4, left). In the example, the developer creates two *target shapes* (one for the fish, one for the line), each represented by one or more nodes. The developer could directly type in 3D coordinates or load a 3D mesh for the bead locations and connections or program their locations (e.g. for algorithmically generated props). However, ArticuLev provides and encourages the use of higher-level tools. For instance, *target shapes* can be picked from a set of parameterized standard targets (e.g., a simple thread with two beads, triangle or square shaped cloth and more). In the example, the line an sinker is defined by selecting a built-in 2-bead `SimpleThreadTarget` and by specifying its length. Developers can also define their own shapes with a 2D point input node. As illustrated in Figure 4 (Left), the creation of the fish simply requires the definition of four beads. In this case, developers can also use an image loader node (Figure 4) to upload a binary target image representing the Fish. This provides a visual reference to specify the beads and ensures target shapes will match only cloth primitives with specific cloth cutouts, as detailed in *Matching detected primitives with target shapes*. The specified local positions should roughly match the positions of the beads on the fabricated prop to ensure robust matching. Of note, it is currently the developer's responsibility to find a reasonable bead arrangement when attaching them to the cloth. Other tools like *LeviProps* [19] can be used independently to verify and automatically adjust the 2D locations of the beads on the cloth so as to fabricate props with optimal trapping stiffness before passing them to ArticuLev.

Given both targets, the `ShapeGroup` node encapsulates them in order to centralize access for the application. Finally, the `ApplicationLogic` contains the developer's C# code for animating the shapes during application runtime.

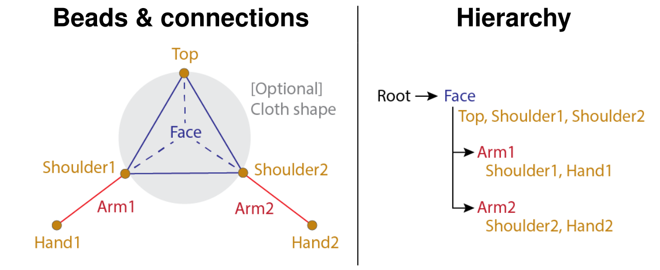

Figure 5. Resulting pipeline input for an articulated shape. Left: The developer visually defines bead positions (yellow), connections and bones (Face, Arm1 and Arm2). Right: Resulting hierarchical representation, with beads assigned to bones (orange), and one Unity node per bone.

The previous example (Figure 4) contained one primitive per target shape (fish prop and thread). Besides such rigid target shapes, developers can also define more complex *articulated* target shapes as shown in Figure 5. To do this, the developer needs to provide a list of bead positions and connections as well as a hierarchy of *bones* (programmatically or in the UI). Conceptually, this is closely related to skeletal computer animation (i.e., a rotating a bone also rotates/translates its child-bones). That is, each bone contains a list of beads, a parent bone and pivot point (bead) to rotate around (e.g., the *Shoulder* beads in Figure 5).

STAGE 1: ANALYZE

This first stage involves no levitation. It consists of the *detection* of the available primitives (using infrared cameras) and the *matching* of those primitives with the target shapes.

Detecting beads and connections

A detection approach for a typical levitation setup must consider the different material properties of the levitation primitives, which means it has to meet the following requirements. First, a multi-pass approach is required, each using different camera settings to detect different materials. Polystyrene beads provide a round and almost Lambertian surface, making them easy to detect even with low exposure values from any viewing angle. However, a longer exposure is required in order to deal with the thinness of threads and the semi-transparency of SuperOrganza (cloth). Second, a multi-camera arrangement is required, to detect thread and cloth primitives. Particularly, thin threads tend to have gaps (non-detected sections) in the individual camera streams. The shininess of SuperOrganza produces bright specular highlights for some viewing angles while reflecting almost no light to other directions. Finally, the primitives are typically not exactly lying on a flat plane (e.g., different bead sizes, thread/cloth bending). Thus, homography solutions cannot be applied directly. Other approaches, such as Steger's accurate line tracing [32] are not applicable due to thread gaps or intermediate beads. All of these challenges and requirements do not directly fit the capabilities of existing commercial software solutions (e.g., *OptiTrack Motive*). Our solution instead uses *OptiTrack*'s low-level *CameraSDK* [20] to directly access and adjust the camera streams, making use of a multi-camera detection algorithm involving three different passes (one per primitive type, each using different camera settings), as described in the following.

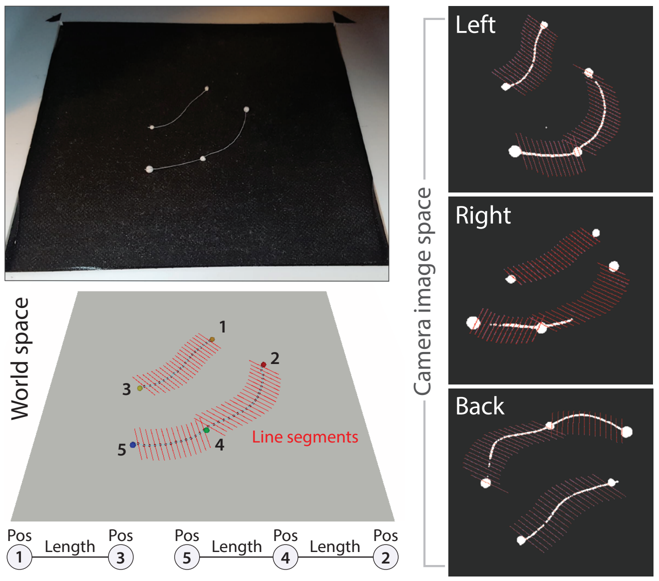

Figure 6. Example for the thread detection algorithm. The algorithm marches between each pair of beads in 3D space while projecting line segments into the camera spaces of each camera. For each projected line segment in the binarized camera images, we count the number of bright pixels along the projected line. As seen in the right column, not all cameras need the thread to be entirely visible for the algorithm to work. The output is an undirected graph with bead positions as nodes and connections including their lengths as edges (bottom left).

Detection pass 1: Bead positions

This pass uses a short camera exposure (65ms in our setup), revealing beads, but not threads or cloth (due to less reflections from their materials). We use *CameraSDK*'s built-in 2D object detection to retrieve circular blobs in each camera stream and unproject them as follows. Given each camera as pinhole model (with their intrinsics and extrinsics), we produce 3D rays that have the camera position as origin and pass through the blob's center in the image plane. Calculating the intersections of those rays from across cameras then gives us the bead locations. Since in practice the rays (created from the blob pixels) do not actually intersect, we use *alglib* [27] to solve the linear equations for approximating the intersection. Given a pair of rays, the calculated optimal solution for those equations is the point that is closest to both rays. The resulting set of 3D bead positions is used as the input to the next passes, which focus on detecting connections between beads.

Detection pass 2: Thread connections

After bead positions from pass 1 are known and stored, the system detects the connections between them. This second pass uses a longer camera exposure, which makes threads appear brighter in the camera streams, but without overexposing (255ms in our setup). In order to deal with potential gaps in the threads within the camera streams (e.g., camera "Right" in Figure 6), this pass needs to combine the inputs received by all cameras to search for connections between each pair of beads. For each pair, we define a *stepping direction* vector that initially points from the first to the second bead to then step in that direction with a fixed step size in world space. At each step, we define a 3D line segment orthogonal to the direction vector (see red lines in Figure 6). We then sample such line segment, projecting its 3D world coordinates into all three camera image spaces (see Figure 6, right). If a bright pixel is hit in any of the camera images, we continue stepping towards the target bead (creating other line segments). Otherwise, the beads are considered not connected. To account for curved threads, we slightly adjust the stepping direction depending on where we find bright pixels along the segment. These connections and the previously computed bead locations produce as undirected graph. The nodes of this graph are bead positions. The edges of the graph indicate thread connections and store the estimated length of the (potentially curved) thread. This graph (e.g., Figure 6, bottom) is later used for matching.

Detection pass 3: Cloth connections

Due to uneven scattering of reflections from cloth, the thresholded images typically contain holes (see Figure 7). Therefore, we combine multiple cameras as with thread detection. If the developer-defined target shapes only feature *convex* cloth primitives (e.g., triangles or squares), then the second detection pass would be enough to identify those as well. However, with arbitrary cloth cutouts (e.g. shapes in Figure 2.Cloth or Figure 9), connections between beads on the same cloth could be missed, e.g., due to intentional holes like the skull's eyes in Figure 9. To also support such *non-convex* shapes, we implemented a simple cloth shape reconstruction from multiple thresholded images, which not only detects beads attached to the same cloth, but also retrieves the fine grained shape of the cloth from the partial views available to each camera. Our shape reconstruction is a variance of the flood fill approach, which additionally takes the 3D position of the attached beads into account. We rasterize a 512×512 pixels top-view of the *ground plane* (16cm × 16cm), with each pixel (u, v) containing a binary value (`Cloth` / `NoCloth`, initially set to `NoCloth`). To initiate a flood fill, we choose an arbitrary (not yet processed) detected bead and start from its position in rasterized image space. To check a pixel (u, v), we first convert it to its x-y-z (right-up-forward) world coordinates. We calculate the x and z coordinates based on the image resolution and the ground plane dimensions ($x = u/512 \cdot 16$, $z = v/512 \cdot 16$). The y-coordinate is estimated as a weighted average of the y coordinates of the beads in the local region. We then perspectively project the world coordinates to pixel coordinates p_{cam} in each camera stream. Finally, if p_{cam} is a bright pixel in any stream, then the rasterization is set to `Cloth` at (u, v) and the flood fill continues from neighboring pixels that have not been visited yet. All beads that are reached within the same flood fill comprise a connected component in the undirected graph.

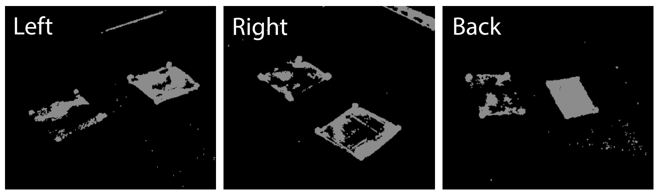

Figure 7. Example for two squared cloth cutouts seen from the three cameras. Due to uneven scattering from the cloth primitives, the thresholded images in each of the independent camera feeds are incomplete, i.e., generally each camera receives different reflections.

Matching detected primitives with target shapes

The detection above produces an undirected graph that incorporates bead positions as well as types and lengths of connections irrespective of the target shapes. ArticuLev *matches* the detected shapes with the developer's input in two separate steps: (1) *Assigning* detected connected components to target shapes and (2) within each assignment, *mapping* detected beads to those in the target shape. Mapping is necessary, because beads are detected and stored in an (undefined) order that generally differs from the developer-defined order in the target shapes. In simple cases, comparing the topologies of the detected connected components and the target shapes primitives is sufficient to find a valid assignment and a mapping. For instance, in the example in Figure 8, it is sufficient to detect, which beads are connected in order to distinguish between the disk (three connected beads) and the two threads (two connected beads each). After assignment, proximity can be used for mapping beads. However, if cloth cutouts as in Figure 9 are used, then different primitives can have the same connection topology even if they represent different shapes. Also, a topology might contain several axis of symmetry, even if the cutout does not. For instance, the fish and the skull in Figure 9 have the same diamond shaped bead topology with rotational symmetries. ArticuLev uses the template image provided by the developer and the detected cloth shape reconstruction for disambiguation. The detected beads are used as anchor points to transform from the provided developer's definition to the shape reconstructed. We iterate over all combinations of mappings between detected and target beads and use singular value decomposition (SVD) [31] to estimate a rigid transformation between both definitions. This usually already discards many bead mappings that cannot be transformed (rigidly) into each other. Among the remaining potential mappings, we compute the overlap of pixels between the transformed reconstruction and the provided template, choosing the bead mapping with the largest overlap (Figure 9, right).

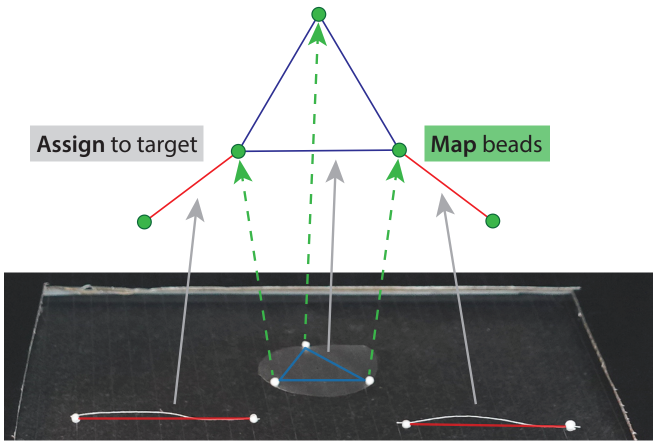

Figure 8. Once beads and connections are detected, ArticuLev matches them with the target shapes. First, ArticuLev *assigns* detected primitives (bottom) to the target shape (top). Second, because beads are detected in an undefined order, ArticuLev needs to *map* the detected beads to the defined target beads.

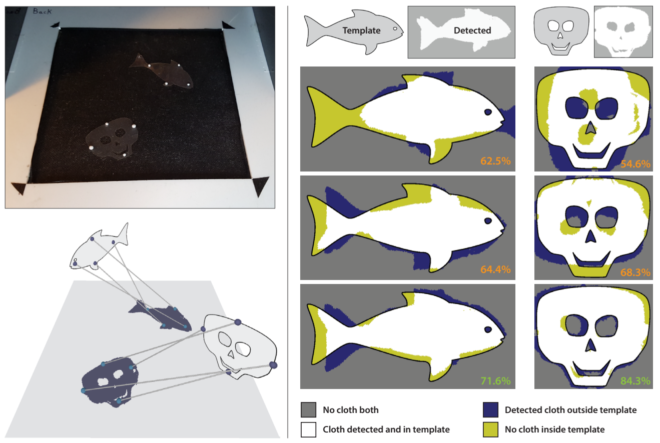

Figure 9. Shape reconstruction and target matching of a fish and a skull cloth cutout. Left: ArticuLev uses the shape reconstruction of the detected cloth primitives to distinguish between target shapes with equivalent topologies. Right: we compute the overlap of the shape reconstructions with the templates to identify the correct target shape (fish versus skull) and correct orientation (rows).

STAGE 2: ASSEMBLE

After all primitives are identified and matched with their target shapes, ArticuLev can start levitating and assembling them accordingly. As shown in Figure 10, this involves three steps: *Lift*, *Join* and *Pose*.

During the *Lift* step, beads in all primitives are levitated vertically to remove contact from the *ground plane*. This step first creates traps at every bead location, smoothly increasing their amplitudes (trapping forces) over an adjustable amount of time (200ms, in our setup) to then lift them a few centimeters. With this, we avoid friction with the ground plane in later steps.

Target shapes can be represented by single props (Figure 4) or by joining multiple primitives (Figure 8). We will discuss the implications for each option in *Single-props versus joining primitives*. If multiple primitives per target are used, then the *Join* step is needed to combine the primitives in mid-air by selectively merging traps. ArticuLev always merges the traps of the primitives horizontally to improve stability. Figure 11 visualizes this for the case of our levitator and the IBP algorithm [15], plotting the stiffness (i.e., trapping strength) of two traps approaching each other horizontally and vertically (top) as well as showing pressure fields of horizontally approaching traps (bottom). This is a symmetric case, i.e., the reported stiffness is the same for each of the two traps. Traps approaching along the vertical axis will interfere destructively, suffering massive drops in stiffness at some points (e.g. $\Delta d = 6mm$). In practice, particles will refuse to move to such points, being instead attracted to points of high stiffness (i.e. peaks at ~ 4.2 or ~ 8.5 mm). In contrast, traps approaching horizontally present a much smoother evolution of stiffness, remaining almost constant for $\Delta d > 8.5$ mm, with a sudden increase at lower distances ($\Delta d < 8.5$ mm). This increase is explained by looking at the evolution of the pressure landscape as traps approach each other (Figure 11, bottom). While traps remain initially separated, at $\Delta d \simeq 8.5$mm these get merged and the beads within will be attracted to each other (i.e. climbing

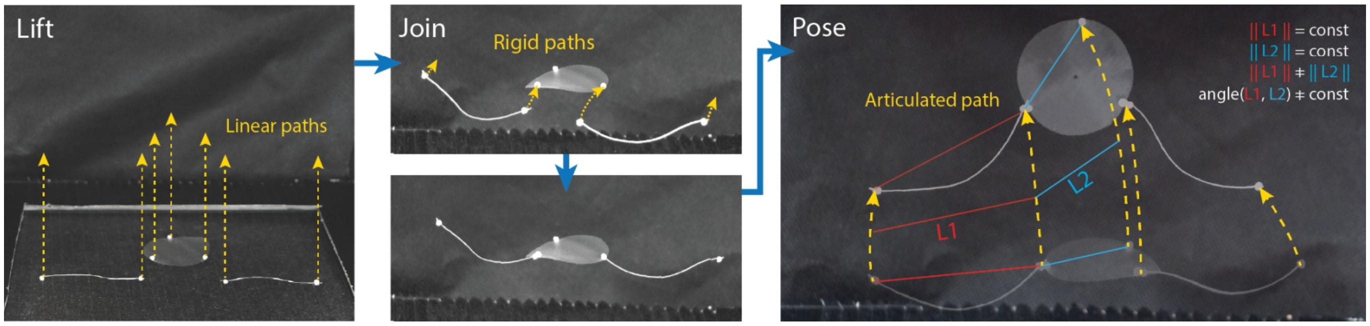

Figure 10. Steps required to assemble target shapes: *Lift* detaches the primitives from the ground plane using linear (upwards) paths. *Join* then assembles them into the required target shape, applying rigid movements and connecting joints along the horizontal direction. *Pose* finally places the shape into the initial pose required by the application logic.

towards $\Delta d \simeq 0$). As a result, the *Join* step merges primitives horizontally while using a relatively high approaching speed (e.g. >0.1m/s in our setup). This avoids the attraction in last stages of merging ($\Delta d < 8.5$ mm) from pulling along connections, which would cause other traps to fail.

Finally, the *Pose* step interpolates the pose of the shape to match their initial target pose, so that the application can start. This typically involves a smooth rotation and translation for each target shape.

Levitation paths

All of the motions required for this stage are supported by *levitation paths*. These allow the position of each trap to be updated smoothly, ensuring small sub-millimeter displacements between each time step to prevent beads from falling between updates. We implemented three generic built-in types of levitation paths:

- `Linear` interpolates all bead positions linearly, with updates not exceeding the maximum step size. This is sufficient to support the *Lift* step in Figure 10.
- `RotationTranslation` transforms a group of beads from an initial to a target position/orientation while maintaining all distances between beads (i.e. rigid-body transformation). The maximum step size is considered in each update for all beads (i.e. the bead furthest from the pivot point dictates how fast the shape can rotate). We use this interpolation, e.g., in the *Join* step in Figure 10.
- `Articulated` generalizes `RotationTranslation` by defining a hierarchy of rigid transformations, required for articulated target shapes. This interpolation maintains relative distances among beads assigned to the same bone, while all updates are still limited by the maximum step size. The *Pose* step in Figure 10 is an example for such a motion, with the beads in each bone (cloth, threads) retaining relative distances within the bone (see L1, L2), but each arm can rotate around the cloth.

We always calculate the rigid motion paths based on the *detected* bead positions on the physical prop, because they usually slightly differ from the local positions described in the target shapes (e.g., beads inaccurately attached to cloth), i.e., the pose, but not necessarily the beads, match the target.

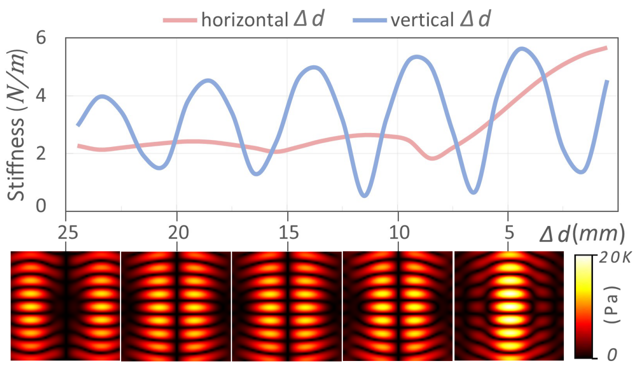

Figure 11. Traps merging in top-bottom levitators. Top: Trapping stiffness varies as traps approach each other along the vertical (blue) or horizontal (red) directions, with the latter providing smoother transitions. Bottom: Pressure fields of horizontally approaching traps. The traps remain separated at longer distances (from the left, 25, 20, 15 and 10 mm), merging as a single trap at shorter distances (right image, 5mm).

STAGE 3: ANIMATE

This stage starts once all primitives have been levitated to physically match the initial state of the target shapes defined by the developers. At this point, a callback function in `ApplicationLogic` (see Figure 4) will be invoked at every frame (60Hz), which developers can use to program their logic. `ApplicationLogic` also provides access to the Unity nodes created for each target shape and bones, which the developer can use to animate the scene. Alternatively, developers can do this by using the levitation paths described earlier, ensuring interpolations comply with maximum step sizes defined. Independently of the method used, ArticuLev automatically updates the underlying traps according to the state of the scene. Since the logic is implemented in Unity and Velt, the developer has access to any existing functionalities, such as input devices like *Leap Motion* or *Kinect* or projection mapping techniques. More information about the communication with the pipeline can be found in the *system architecture* appendix.

Figure 12 illustrates a concrete envisioned application - a levitation-based teleconferencing system. The application combines high resolution projection mapping with a physicalized way of expressing the body language of the remote participant. This example uses a disk-shaped piece of cloth and two threads to form an articulated shape. The logic is

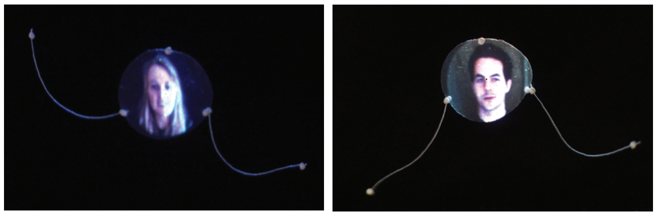

Figure 12. A teleconferencing example based on ArticuLev. A levitated disk serves as projection surface to display the users' faces. The threads reflect the users' gestures to physicalize their body language.

programmed to manipulate the bones of the articulated target shape to match the arms, as tracked with a *Kinect* device (e.g. threads can freely rotate to convey hand/arm gestures). Velt handles the projection mapping and the acquisition of the RGB-D stream from *Kinect*.

Single-props versus joining primitives

A target shape in a running ArticuLev application can be physically represented either as single pre-assembled prop or by joining multiple primitives in mid-air (*Join* step in assembly). For instance, instead of assembling the the target shape in Figure 10 and Figure 12 from three primitives, the arm threads could instead be directly sewed into the beads of the cloth. Figure 13 shows the general difference. While this is equivalent for the target shapes (ArticuLev will either pick up a single prop or join multiple primitives, as long as they yield the same target shape in the end), this does affect the behavior of the primitives during articulated movements during animation.

The first option (single pre-assembled prop) generally requires less traps during the assembly process (allowing stronger traps) and could be visually more appealing for some cases, as strictly only one bead is located in each trap. However, this connection (thread sewn onto bead) can constraint the articulation, adding tension to the bead as the arm is moved and hence can limit the possible animations. In the second option (joining primitives), producing target shapes by joining simple primitives (e.g. triangles, threads) can allow reusing a small set of primitives

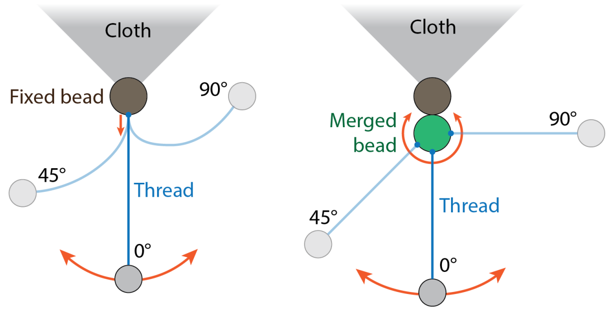

Figure 13. Target shapes can be physically displayed by fabricating pre-assembled props (left) or by joining multiple primitives into a bigger shape (right). In both cases, the "Fixed bead" (brown) cannot rotate relative to the cloth it is attached to. In a pre-assembled shape (left: thread directly sewed into brown bead), rotating the bead at the lower end of the thread causes unwanted bending forces on the thread. When joining separate primitives (right: separate thread primitive with two beads), beads share a trap, but retain their degrees of freedom (see green "Merged bead"). This allows the thread primitive to rotate independently of the cloth, making articulated movements more stable.

to produce a range of target shapes. More importantly, beads sharing a trap will retain their degrees of freedom in terms of rotation, removing constraints from the bones, which can improve stability (e.g., arm rotations Figure 12).

While the application logic remains the same, these differences in behavior should be taken into account when fabricating the props and primitives. The differences are illustrated in motion in *Video Figure A* (in supplemental material).

TECHNICAL EVALUATION

We conducted an in-lab evaluation, in order to characterize the capabilities of ArticuLev across each of its different stages and for different combinations of levitation *primitives*. More specifically, we designed six test applications that make use of all primitive materials individually and in pairs (in Table 1 from left to right: bead-only, bead-thread, bead-cloth, thread-thread, thread-cloth, cloth-cloth). Our tests strictly only run *one* cycle per trial to measure the success rate. Our real-time implementation could trivially identify and correct failure cases, by restarting the pipeline (i.e., introducing only slight delays in practice). However, our strict one-cycle tests were used as means to identify the causes of failure and to propose further improvements to both software (stages) and primitive materials (see *Limitations and future work*). We conducted the tests in groups of 10 trials per application, collected across 3 consecutive days (i.e. varying lighting conditions, system calibrated at the beginning of each day), resulting in 30 collected trials per test application. Each trial assessed ArticuLev's capability to detect and assemble the primitives required, and it involved two operators - the experimenter and a non-expert operator (i.e., no computer science, research or engineering background). The non-expert placed the required primitives onto the levitator's ground plane, out of sight of the experimenter, who then started ArticuLev and registered the results. The non-expert operator could place the props arbitrarily, but was made aware of the working volume of the device. Furthermore, the non-expert was instructed to place primitives with no overlap and with a minimum distance of $\sim$ 25mm between beads (see *Hardware and software*).

Table 1 shows the results of our evaluation and serves as the reference table throughout this section. We split our results in two groups: *Primitive success rates* and *Stage success rates*.

With the *Primitive success rates*, we report the rate of success over all individual primitives *across* trials regardless whether the stage was successful. Detection in primitive success rates are the percentages of primitives, which were correctly identified and matched to target shapes. For instance, detecting 30 out of 60 individual bead primitives across all trials would mean 50% primitive success rates. Similarly, we also report the overall number of individual primitives that physically reach their target shape. *Assembly* reports the average and standard deviation in relation to the total number of primitives across trials within each test application.

With *Stage success rates*, we describe the rate of *complete* success *within* a pipeline step. For instance, detecting all beads in an application counts towards the success, whereas missing just one bead does not count towards success. Analogously

	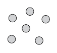	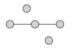	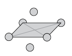	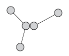	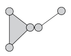	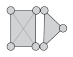
Beads	6	5	6	5	5	7
Connections	0	2	6	3	4	9
Primitive success rates %						
Detection	98	100	100	100	100	98
Assembly	93 ± 10	83 ± 17	98 ± 8	76 ± 28	83 ± 23	82 ± 27
Stage success rates %						
Detect: Beads	90	100	100	100	100	93
Connections	N/A	100	93	100	100	93
Assembly: Lift	70	83	93	74	77	80
Join+Pose	66	50	63	57	66	66

Table 1. Summary of results obtained from our evaluation. Columns represent each of the target shapes tested, while the rows represent shape input parameters (beads and connections) and measured parameters for each stage. All percentages in the stage success rates are in relation to the total amount of trials independent of the success of the previous stage, i.e., they can only remain equal or get lower in subsequent stages.

in assembly, trials in which *any* of the primitives failed to reach their target positions are counted as unsuccessful. For instance, in the cloth-thread example (5th from the left) if the thread *or* the cloth failed, it does not count towards the success rate. All percentage entries throughout the stages are always in relation to the total amount of primitives of the respective test application. That is, a failure in a step counts as a failure in later steps. For instance, a bead that is not detected is also not lifted. Hence, percentages can only remain the same as the previous stage in best case (no further failures in the step) or get lower. This also implies that the last row of the stage success rates (*Join+Pose*) shows the percentage of trials, which where completely successful, i.e., all the way from detection to reaching the target pose. In the following, we discuss the stage success rates in more detail.

For the *Detection* stage, we tested the number of trials, in which all beads and connections were correctly identified by the system. We further separated this stage into the detection of the *Beads* and their *Connections*. This stage shows robust results, with high detection rates for all beads and connections. The few failures were related to incomplete reconstructions for cloth primitives (i.e., particularly when placed at the limits of the working volume, with worse illumination), or for primitives placed too closely together (i.e., beads rolling towards other primitives after placement). No false positives (non-existing beads or connections) were produced by the system.

We split the *Assembly* stage test into two steps: *Lift* and *Join+Pose* (whereas only the last three test applications include a *Join* step). A large part of the assembly failures occurred during the *Lift* step, usually due to primitives sticking to the ground plane (i.e., micro-fibers in primitives or felt, electrostatic attraction due to friction). Other errors within *Assembly* were caused by primitives interacting during the joining or posing steps, either by directly hitting them or by distorting nearby traps holding them (e.g. a bead going near/through a thread primitive). Thus, collision avoidance mechanisms for levitation [28] should be considered to avoid these cases. In other cases, failures could be related to specifically ill-posed arrangements of the traps (some arrangements lead to destructive interference [19]), which can be alleviated using alternative levitation algorithms (e.g., IBP [15] or GS-PAT[12]).

Evaluation summary

We evaluated the ArticuLev pipeline with six target combinations with our specific setup. The overall results vary according to the test application, while the detection is close to 100%, only 50% - 66% of the cases successfully reach the final stage. Our system is agnostic to the low-level trapping algorithm and the number of cameras. In general, adding more cameras and hence, more redundancies would further improve the detection. However, from the results, we can conclude that three cameras with thresholded streams are already sufficient to reliably detect beads and connections with commonly used levitation materials. In particular, we can confirm that with the reflection properties of organza, generally at least one out of three cameras receive enough light to detect the cloth, even when placed at different orientations. At the same time, beads that are attached to organza can still be detected by all cameras as those are still brighter than the organza reflections from all angles. While low-level improvements for trap generation are possible, the overall ArticuLev pipeline with our used setup was able to persistently detect levitation primitives and generate valid assembly sequences.

EXAMPLE COMBINATIONS OF PRIMITIVES

We demonstrate the potential of ArticuLev by implementing a number of levitation-based prototypes (see Figure 14). All examples were created using Velt nodes (see Figure 5) and simple application logic. They showcase either individual primitives or combinations primitives with of different materials (e.g., bead-thread, thread-cloth) and/or display approaches (static or fast moving PoV content), showcasing ArticuLev's unique ability to support heterogeneous levitated interfaces. We invite the reader to see all prototypes in motion in the supplemental video of this publication (main video).

The first three examples (Figure 14.A) illustrate single cloth primitives The airplane (Left) shows a flat prop, while the top and bottom beads in the skull (Center) are slightly bent for a curved prop appearance. The handcuffs (Right) illustrate an articulated, single cloth shape (i.e., one bone per ring, another for the connecting chain), allowing independent constrained manipulation of each part. This example is similar to those supported by LeviProps [19], i.e., their algorithm can be used to optimize the bead placement when fabricating this type of cloth-based props. However, as opposed to LeviProps, our system can uniquely identify the correct orientation when posing the props and our approach is robust in terms of inaccurately placed beads or differently sized shapes. The next two examples (Figure 14, B and C) illustrate the use of articulated thread primitives, and the two alternatives to create articulations (single prop or joined primitives, as in Figure 13). In Figure 14.B, one thread with three beads and one thread with two beads are joined and animated to reflect the movements of the user's index and thumb fingers (tracked by *Leap*

Motion). In contrast, Figure 14.C shows an articulated stick figure made of a single pre-assembled prop, animated with a mouse. The subsequent examples (Figure 14, D, E and F) illustrate combinations of different types of primitives and/or display approaches. Figure 14.D combines cloth and thread to represent a fish and fishing line, animated independently (e.g., swimming fish animation). Figure 14.E combines three pieces of cloth into one articulated shape, creating a flying skull with a wing flap animation. Finally, (Figure 14.F shows a thread and a fast-moving bead creating a circle at PoV rates. In addition, a slow sinusoidal rotation is applied to both the thread and bead, to produce an oscillating pendulum. Finally, Figure 14.G illustrates one example for the incorporation of Unity functionalities - the physics engine applied to a square cloth primitive. The motion of the square mimics that of a rigid body, bouncing off the ground plane and invisible walls around the levitator's working volume.

While the built-in ArticuLev functionalities were sufficient to produce the prototypes that we described in this section, ArticuLev can be extended programmatically, so as to support specialized shapes and assembly sequences (e.g., folding). More details about the modular pipeline including a more complex example can be found in the *system architecture* appendix and in *Video Figure B* (both in supplemental material).

LIMITATIONS AND FUTURE WORK

While the pipeline provides a self-contained system enabling novel types of levitated applications, its individual components can be improved and further generalized in the future. Our evaluation showed how the inclusion of collision avoidance strategies [28] to the *Assemble* stage could prevent some failure cases. In addition, adopting recent levitation algorithms like GS-PAT [12] could further increase robustness and provide full PoV content support (i.e. performance should be increased to update the traps at >10K updates per second). Another possible future directions would be the disassembly of primitives to morph between shapes in mid-air.

Further improvements could be attained by exploring alternative material choices for the levitation primitives. Some *Lift* failures were caused by tangling micro-fibres from thread primitives (cotton) and the ground plane (felt). Electrostatic effects can also hold the primitives (i.e., both polystyrene and Organza produce triboelectric effects), and triboelectric-neutral materials should be considered. The optical properties (e.g. light scattering) of such alternative materials should also be revisited and may require adaptations to the detection pipeline. Similarly, using other levitator arrangements (e.g. four boards, as in *PixieDust* [23]) not only changes trap topologies and assembly strategies when merging traps (see Figure 11), but might also affect safety during usage.

Even if ArticuLev can facilitate initialization and creation of levitated experiences (e.g. visual nodes, Unity integration), its target group still focuses on developers and might not be suitable for content creators without any programming background. During development, we remotely collaborated with an artist to create an interactive experience, which is as a first step to bring in relevant perspectives and facilitate further adoption of levitation experiences.

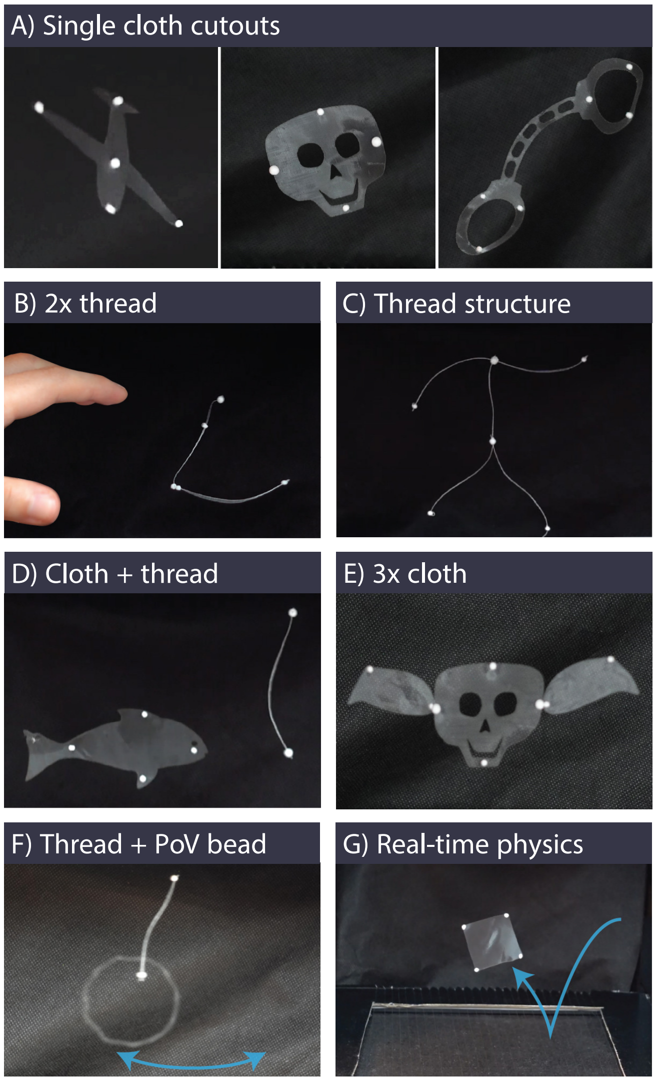

Figure 14. Example prototypes created with ArticuLev to showcase the supported primitives and heterogeneous primitive combinations.

CONCLUSION

We presented the integrated detection and levitation pipeline ArticuLev, which unifies various levitation approaches from previous work and enables novel combinations of shape primitives. We tackle a problem that is unique to levitation based mid-air displays, as those rely on external props and their assembly to be able to display content. Our pipeline detects connections between beads with an approach that is carefully designed around the requirements of a typical levitator setup and the materials of shape primitives. Furthermore, with our examples, we demonstrated how our pipeline generalizes previous levitation-based mid-air displays and allows novel combinations of shape primitives. In addition, with our shape assembly approach, we present the first acoustic levitation system to assemble animated fully articulated shapes. We believe that our pipeline and formalization are an important step to allow effective application development for acoustic levitation based mid-air displays.

ACKNOWLEDGEMENTS
This work was supported by the EU's H2020 ERC Advanced Grant (787413), ERC Proof of Concept grant (841781), the Royal Academy of Engineering Chairs in Emerging Technology Program, and the AHRC UK-China Research-Industry Creative Partnerships AH/T01136X/2. We thank Eimontas Jankauskis for creating the stylizations used in Figure 13 and in *Video Figure A*.

REFERENCES
[1] Myroslav Bachynskyi, Viktorija Paneva, and Jörg Müller. 2018. LeviCursor: Dexterous Interaction with a Levitating Object. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces (ISS '18)*. Association for Computing Machinery, New York, NY, USA, 253–262. DOI: http://dx.doi.org/10.1145/3279778.3279802

[2] Johann Berthelot and Nicolas Bonod. 2019. Free-space micro-graphics with electrically driven levitated light scatterers. *Opt. Lett.* 44, 6 (Mar 2019), 1476–1479. DOI: http://dx.doi.org/10.1364/OL.44.001476

[3] EH Brandt. 1989. Levitation in physics. *Science* 243, 4889 (1989), 349–355.

[4] Elan Dubrofsky. 2009. Homography estimation. *Diplomová práce. Vancouver: Univerzita Britské Kolumbie* (2009).

[5] Andreas Fender and Jörg Müller. 2018. Velt: A Framework for Multi RGB-D Camera Systems. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces (ISS '18)*. Association for Computing Machinery, New York, NY, USA, 73–83. DOI: http://dx.doi.org/10.1145/3279778.3279794

[6] Euan Freeman, Asier Marzo, Praxitelis B. Kourtelos, Julie R. Williamson, and Stephen Brewster. 2019. Enhancing Physical Objects with Actuated Levitating Particles. In *Proceedings of the 8th ACM International Symposium on Pervasive Displays (PerDis '19)*. Association for Computing Machinery, New York, NY, USA, Article Article 2, 7 pages. DOI: http://dx.doi.org/10.1145/3321335.3324939

[7] Euan Freeman, Julie Williamson, Sriram Subramanian, and Stephen Brewster. 2018. Point-and-Shake: Selecting from Levitating Object Displays. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, Article Paper 18, 10 pages. DOI:http://dx.doi.org/10.1145/3173574.3173592

[8] T. Fushimi, A. Marzo, B. W. Drinkwater, and T. Hill. 2019. Acoustophoretic volumetric displays using a fast-moving levitated particle. *Applied Physics Letters* 115 (2019), 064101.

[9] Ryuji Hirayama, Diego Martinez Plasencia, Nobuyuki Masuda, and Sriram Subramanian. 2019. A volumetric display for visual, tactile and audio presentation using acoustic trapping. *Nature* 575, 7782 (2019), 320–323.

[10] Jinha Lee, Rehmi Post, and Hiroshi Ishii. 2011. ZeroN: Mid-Air Tangible Interaction Enabled by Computer Controlled Magnetic Levitation. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. Association for Computing Machinery, New York, NY, USA, 327–336. DOI:http://dx.doi.org/10.1145/2047196.2047239

[11] Mark Marshall, Thomas Carter, Jason Alexander, and Sriram Subramanian. 2012. Ultra-Tangibles: Creating Movable Tangible Objects on Interactive Tables. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2185–2188. DOI: http://dx.doi.org/10.1145/2207676.2208370

[12] Diego Martinez-Plasencia, Ryuji Hirayama, Roberto Montano-Murillo, and Sriram Subramanian. 2020. GS-PAT: High-Speed Multi-Point Sound-Fields for Phased Arrays of Transducers. *ACM SIGGRAPH* (2020). (To appear).

[13] Asier Marzo. 2016. GauntLev: A Wearable to Manipulate Free-Floating Objects. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3277–3281. DOI: http://dx.doi.org/10.1145/2858036.2858370

[14] A. Marzo, T. Corkett, and B. W. Drinkwater. 2018. Ultraino: An Open Phased-Array System for Narrowband Airborne Ultrasound Transmission. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 65, 1 (2018), 102–111.

[15] Asier Marzo and Bruce W Drinkwater. 2019. Holographic acoustic tweezers. *Proceedings of the National Academy of Sciences* 116, 1 (2019), 84–89.

[16] Asier Marzo, Sue Ann Seah, Bruce W Drinkwater, Deepak Ranjan Sahoo, Benjamin Long, and Sriram Subramanian. 2015. Holographic acoustic elements for manipulation of levitated objects. *Nature communications* 6 (2015), 8661.

[17] Asier Marzo, Sriram Subramanian, and Bruce W. Drinkwater. 2017. LeviSpace: Augmenting the Space above Displays with Levitated Particles. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. Association for Computing Machinery, New York, NY, USA, 442–445. DOI: http://dx.doi.org/10.1145/3132272.3132295

[18] G. Memoli, M. Caleap, Michihiro Asakawa, D. R. Sahoo, B. Drinkwater, and Sriram Subramanian. 2017. Metamaterial bricks and quantization of meta-surfaces. *Nature Communications* 8 (2017).

[19] Rafael Morales, Asier Marzo, Sriram Subramanian, and Diego Martínez. 2019. LeviProps: Animating Levitated Optimized Fabric Structures Using Holographic Acoustic Tweezers. In *Proceedings of the 32nd Annual*

ACM Symposium on User Interface Software and Technology (UIST '19). Association for Computing Machinery, New York, NY, USA, 651–661. DOI: http://dx.doi.org/10.1145/3332165.3347882

[20] NaturalPoint. 2020a. Camera SDK. (2020). https://optitrack.com/products/camera-sdk/ (Accessed: 02/01/2020).

[21] NaturalPoint. 2020b. OptiTrack. (2020). https://optitrack.com/ (Accessed: 02/01/2020).

[22] Mohd Adili Norasikin, Diego Martinez, Spyros Polychronopoulos, Gianluca Memoli, Yutaka Tokuda, and Sriram Subramanian. 2018. SoundBender: Dynamic Acoustic Control Behind Obstacles. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. Association for Computing Machinery, New York, NY, USA, 247–259. DOI:http://dx.doi.org/10.1145/3242587.3242590

[23] Yoichi Ochiai, Takayuki Hoshi, and Jun Rekimoto. 2014. Pixie Dust: Graphics Generated by Levitated and Animated Objects in Computational Acoustic-Potential Field. *ACM Trans. Graph.* 33, 4, Article Article 85 (July 2014), 13 pages. DOI: http://dx.doi.org/10.1145/2601097.2601118

[24] Themis Omirou, Asier Marzo, Sue Ann Seah, and Sriram Subramanian. 2015. LeviPath: Modular Acoustic Levitation for 3D Path Visualisations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 309–312. DOI:http://dx.doi.org/10.1145/2702123.2702333

[25] T. Omirou, A. M. Perez, S. Subramanian, and A. Roudaut. 2016. Floating charts: Data plotting using free-floating acoustically levitated representations. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. 187–190. DOI: http://dx.doi.org/10.1109/3DUI.2016.7460051

[26] Viktorija Paneva, Myroslav Bachynskyi, and Jörg Müller. 2020. Levitation Simulator: Prototyping Ultrasonic Levitation Interfaces in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. DOI:http://dx.doi.org/10.1145/3313831.3376409

[27] ALGLIB Project. 2020. (2020). https://www.alglib.net/ (Accessed: 23/06/2020).

[28] Maxime Reynal, Euan Freeman, and Stephen Brewster. 2020. Avoiding Collisions When Interacting with Levitating Particle Displays. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–7. DOI: http://dx.doi.org/10.1145/3334480.3382965

[29] Deepak Ranjan Sahoo, Takuto Nakamura, Asier Marzo, Themis Omirou, Michihiro Asakawa, and Sriram Subramanian. 2016. JOLED: A Mid-Air Display Based on Electrostatic Rotation of Levitated Janus Objects. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. Association for Computing Machinery, New York, NY, USA, 437–448. DOI: http://dx.doi.org/10.1145/2984511.2984549

[30] D. Smalley, E. Nygaard, K. Squire, J. V. Wagoner, J. Rasmussen, S. Gneiting, K. Qaderi, J. Goodsell, W. Rogers, M. Lindsey, K. Costner, A. Monk, M. Pearson, B. Haymore, and J. Peatross. 2018. A photophoretic-trap volumetric display. *Nature* 553 (2018), 486–490.

[31] Olga Sorkine. 2009. Least-squares rigid motion using svd. *Technical notes* 120, 3 (2009), 52.

[32] Carsten Steger. 1998. Evaluation of subpixel line and edge detection precision and accuracy. *International Archives of Photogrammetry and Remote Sensing* 32 (1998), 256–264.

[33] W Le Conte Stevens. 1899. A Text-Book of Physics–Sound. (1899).

[34] Ultraleap. 2020. Ultraleap STRATOS. (2020). https://www.ultraleap.com/product/stratos-explore/ (Accessed: 04/01/2021).