Tactile Animation by Direct Manipulation of Grid Displays

Oliver S. Schneider[1,2]
oschneid@cs.ubc.ca

Ali Israr[1]
israr@disneyresearch.com

Karon E. MacLean[2]
maclean@cs.ubc.ca

1. Disney Research, Pittsburgh, USA 2. University of British Columbia, Vancouver, Canada

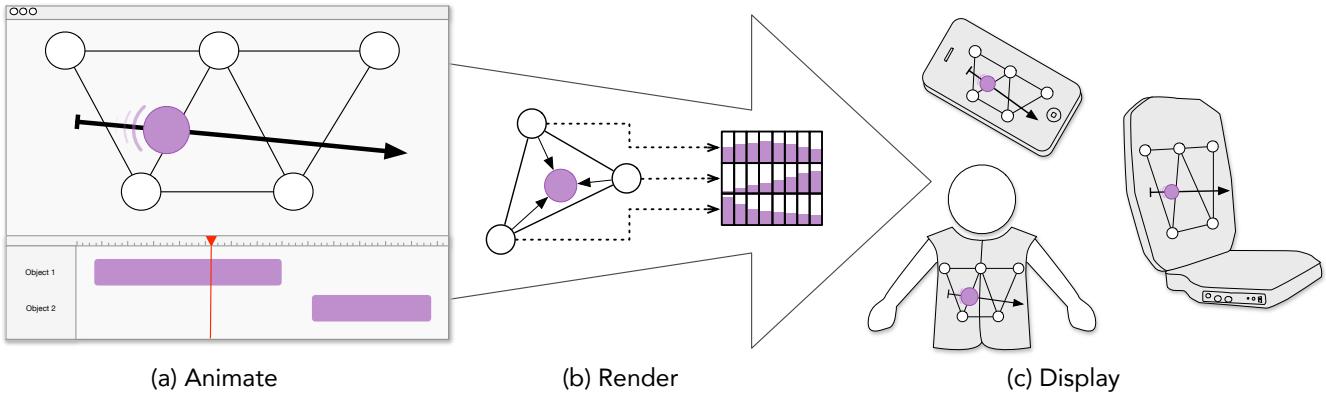


Figure 1: Concept sketch for tactile animation. An artist draws an animated sequence in the user interface and the user experiences phantom 2D sensations in-between discrete actuator grids. The animator controls phantom sensations directly, feeling the sensation in real-time to design expressive sensations for arbitrary vibrotactile arrays.

ABSTRACT

Chairs, wearables, and handhelds have become popular sites for spatial tactile display. Visual animators, already expert in using time and space to portray motion, could readily transfer their skills to produce rich haptic sensations if given the right tools. We introduce the *tactile animation object*, a directly manipulated phantom tactile sensation. This abstraction has two key benefits: 1) efficient, creative, iterative control of spatiotemporal sensations, and 2) the potential to support a variety of tactile grids, including sparse displays. We present Mango, an editing tool for animators, including its rendering pipeline and perceptually-optimized interpolation algorithm for sparse vibrotactile grids. In our evaluation, professional animators found it easy to create a variety of vibrotactile patterns, with both experts and novices preferring the tactile animation object over controlling actuators individually.

Author Keywords

Haptics; vibrotactile; animation; design.

ACM Classification Keywords

H.5.1. Multimedia Information Systems: Animations; H.5.2. User Interfaces: Haptic I/O

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
UIST'15, November 08–11, 2015, Charlotte, NC, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN/15/11...$15.00.
DOI: http://dx.doi.org/10.1145/2807442.2807470

INTRODUCTION

Haptic feedback is viewed today as a key ingredient of immersive media experiences. Body-moving devices in theatre seats, ride vehicles, and gaming platforms can tilt, translate, and shake the user for increased engagement. Recently, arrays of multiple actuators have been developed to display expressive, spatial sensations on the skin [4, 13, 17, 30, 37].

Vibrotactile (VT) arrays, which stimulate the skin through vibration, are common in diverse applications from immersive gaming chairs [13] to wearable vests for mobile awareness [15]. These displays typically employ sparse actuator arrangements to reduce cost and power requirements, using perceptual illusions to create continuous sensations [1,12,29]. Unfortunately, adoption of VT arrays is limited by a lack of authoring tools. Most only support a single actuator [6]; those that accommodate multiple actuators control each separately [17, 23, 32], cumbersome for non-adjacent actuators.

To remedy this, we propose the *tactile animation object*, an abstract, directly manipulable representation of a phantom sensation perceived in-between physical actuators. With this approach, designers can efficiently and creatively explore ideas and iterate without worrying about underlying actuator arrangements. As long as a rendering algorithm can be developed, this abstraction not only facilitates design, but is compatible with a variety of form factors and technologies.

In this paper, we describe the tactile animation object and implement it in *Mango*, a tactile animation tool and pipeline (Figure 1). Our contributions are: 1) A tactile animation interface grounded in user interviews and prior literature. 2) A rendering pipeline translating tactile animation objects to

phantom sensations on sparse, generalized VT arrays, optimized with a perceptual study. 3) An evaluation with professional animators showing accessibility and expressivity. 4) An exploration of potential applications for tactile animation.

BACKGROUND

Haptic Entertainment Technologies

Haptic feedback was used in cinema as early as *Percepto*, a 1959 multisensory experience for the movie "The Tingler" [11] with theater seats that buzzed the audience at strategic moments. Current 4D theaters, rides, shows, and gaming arcades are equipped with sophisticated motion platforms (e.g., D-Box, www.d-box.com) that supplement visual scenes. Large tactile transducers (such as Buttkickers, www.thebuttkicker.com) that shake the entire seat using the sound stream are also common with gaming and music content. Custom editors (such as D-Box Motion Code Editor) and software plugins overlay visual and audio content with haptics, and allow designers to generate, tune and save frame-by-frame haptics in an allocated track.

In contrast to displacing the entire body, multichannel haptic devices create percepts of dynamic and localized haptic sensations on the user's skin [13] and in mid-air [37]. Similar devices have been developed for online social interactions using custom multi-actuator displays [17,23,35]. All of these technologies require extensive programming experience, knowledge of hardware and background in haptic sciences to generate expressive and meaningful haptic content. Without guiding principles or haptic libraries, content generation schemes are complex, device-specific, and time consuming.

Another class of haptic technology renders high-resolution spatio-temporal patterns on the skin using a sparse array of VT actuators. These technologies use parametric models of sensory illusions in touch, such as phantom tactile sensations [1], and create illusory vibrations in between two or more VT actuators. This idea has been used to create a perceived motion flow between two vibrators mounted on the ends of a handheld device [29] and to create across-the-body and out-of-the-body illusions on a mobile device using up to four actuators [20]. The Tactile Brush algorithm [12] combined phantom tactile sensations and apparent tactile motion to render high-resolution and moving haptic patterns on the back using a coarse grid of VT actuators, but paths must be pre-determined (Figure 2a). Other spatio-temporal VT illusions such as the "cutaneous rabbit" [33] and Tau and Kappa effects [8] can be also used with VT arrays.

Haptic Authoring Tools

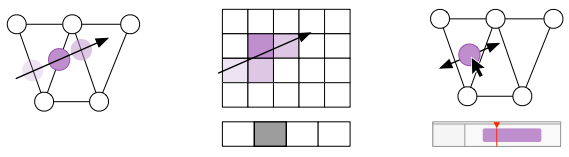
As long as designers have considered haptic effects for entertainment media, they have needed compositional tools [7]. Requirements drawn from previous work on how to prototype, sketch, or control haptic phenomena using non-programming methods are summarized in Table 1.

The Hapticon editor [6], Haptic Icon Prototyper [31], posVibEditor [25], and Immersion's Haptic Studio (www.immersion.com) use graphical representations to edit either waveforms or profiles of dynamic parameters (such as frequency or torque) over time. Another approach is predefining a library of haptic patterns to augment media content. Immersion Corporation's Touch Effects Studio lets users enhance a video from a library of tactile icons supplied on a mobile platform. Vivitouch Studio [32] allows for haptic prototyping of different effects alongside video (screen captures from video games) and audio. These tools focus on low-level control of device features rather than a semantic space, and control devices with either a spatial or temporal component, but not both simultaneously.

Several tools have allowed users to author haptic content using accessible touchscreen interactions. A demonstration-based editor [10] allowed control of frequency and intensity by moving graphical objects on a screen. mHIVE [26] controls frequency, intensity, waveform and envelope of two tactors with touchscreen gestures. Both systems were shown to be intuitive and easy to use for exploration or communication, but faltered when refining more elaborate sensations. Commercially, Apple's vibration editor (since iOS 5, 2011) allows users to create personalized vibratory patterns by touching the screen, but only produces binary on/off timing information.

Other aids to creating haptic phenomena include haptic sketching [21] for hands-on exploration of haptic ideas in early design, and end-user customization of tactile sensations [28]. Both emphasize exploration and broad manipulation rather than finely controlled end results. HAMLAT [5] supports authoring of force feedback in static 3D scenes. Lee and colleagues [18] used a musical metaphor for vibrotactile authoring. Schneider et al. introduced "FeelCraft" for end user customization of a library of *feel effects* [27].

Kim and colleagues offered combined spatial and temporal control using a tactile video metaphor for dense, regular arrays of tactile pixels ("taxels"), including a feature of sketching a path on video frames [17] (Figure 2b). While a promising approach, this tool relies on editing of discrete actuators and frames, with its sketching feature used for input, not as a manipulation method. As well, it does not generalize to sparse or irregular displays, and was not evaluated with designers. We suggest that an animation metaphor could provide an easier interaction model, facilitating key creative activities such as rapid exploration and iteration, especially through a continuous timeline (Figure 2c). The control of multi-actuator outputs has also been explored by TactiPEd [23] and Cuartielles' proposed editor [3]. However, these approaches still require the separate control of different actuators, rather than a single perceived sensation produced by the multi-actuator device.



(a) Tactile Brush [12]: precomputed paths (b) Tactile Video [17]: frames of tactile pixels (c) Tactile Animation: direct manipulation

Figure 2: Comparison between related systems.

LR	Description
LR1	**Real-Time Playback** [21, 26] Rapid prototyping is essential for working with VT sensations, especially in absence of objective metrics. Feeling a sensation at design time allows iteration to converge faster to better results. However, *too* real-time can cause split attention.
LR2	**Load, save, manipulate** [14, 24, 26] A persistent object model is essential for sensation editing over longer projects and sharing with other designers or across devices. Well-defined actions upon a data structure also facilitates features like *undo* that support experimentation.
LR3	**Library of effects** [6, 9, 23, 31, 32] A library of saved sensations is an important feature used in previous haptic authoring tools, providing inspiration and preventing designers from re-inventing the wheel.
LR4	**Device configuration** [17–19, 23] Because of the many types of haptic devices, a general tool must be able to understand different devices. Lightweight configuration files are common in the literature, allowing users to select specific hardware, specify location and type of actuators, and choose a rendering algorithm.
LR5	**Multiple channels & combination of effects** [6, 23, 25, 31, 32] Being able to display multiple effects simultaneously, or combine effects via superposition or concatenation, is essential for expanding the design space. This is typically represented in a timeline, which represents the temporal behaviour of any objects.
LR6	**Visual/direct control metaphor** [3, 17, 23] Most previous tools consider each actuator separately. When thinking semantically about a spatial system, a direct view of the device and actuator layout is critical for direct manipulation.
LR7	**Audio/visual context** [17, 21, 32] Haptic perception depends greatly on additional senses [8]. By providing audio and visual feedback, these effects can be mitigated and the designer can experience haptic sensations in context.
LR8	**User Feedback** [26, 32] Receiving feedback from users, either by demonstration or A/B testing, is extremely valuable.

Table 1: **Literature Requirements (LRs)** for a tactile animation authoring tool.

TACTILE ANIMATION AUTHORING TOOL

Our objective is to provide media designers with a familiar and efficient framework for creating dynamic haptic content. Mango's design is based on two sets of requirements: Literature ("LRs", Table 1), from prior research on haptic authoring tools, and Industry ("IRs") from interviews with five industry experts in haptic media creation and animation, which confirm and expand upon design decisions for other VT tools.

Gathering Design Requirements

We interviewed two industry experts with haptics experience from a media company (E1-2). E1 uses Max/MSP, OpenFrameworks, Processing, and Visual Studio to create haptic media. E2 is a professional media designer and an expert user of Pro Tools (an industry standard for authoring sound media). Together, E1 and E2 previously undertook a six-month training that included generation of dynamic haptic experiences on seats and supporting platforms using audio and video tools. Our interviews included meetings, recordings, and sketches of their experience during training.

In addition, we conducted contextual interviews of three industry animators (A1-3) interacting with non-tactile animation tools using a think-aloud protocol. A1 and A3 used Adobe After Effects, while A2 used Maya. A1 and A2 were tasked with creating an animation of two balls moving; A3 created an animation based on a sound file. These interviews yielded rich detail that we compiled into categories, then compared with our LRs (Table 1). LRs 2-7 also emerged independently from this stage. We extend the LRs with additional expert-drawn **industry requirements (IRs):**

IR1 - Animation window allows users to draw tactile animation objects, control them in space, and define their motion paths. The window is overlaid with location and type of haptic actuators, providing visual feedback (LR8).

IR2 - Timeline is a time track for a tactile animation object. During playback, the animation is played on IR1 showing the movement of the animation relative to the tactile object. Object behaviours are linked to time track to visualize temporal variations. Time tracks are editable by inserting key frames.

IR3 - Object tools extend LR2, supporting direct manipulation operations on tactile objects such as "new", "scale", "translate", analogous to object creation and manipulation in After Effects and Maya.

IR4 - Path tools define motion paths of tactile objects (straight lines, curves, input-device traces), and store them in a path library (LR3).

IR5 - Haptic rendering schemes compute output waveforms for each actuator channel, animated visually in the animation window. Users select the scheme from a list for connected hardware, defined in a hardware configuration file (LR4).

IR6 - Global parameter tools allow the user to control the overall feel of the tactile animation object. Analogous to filters and effects applied on the object, this includes parameter setting for frequency, intensity and modulation.

We developed a tool design from these two sets of requirements. Our Mango prototype uses Python 2.7 and Tkinter for the rendering pipeline (Figure 3) and UI (Figure 4), which communicates with haptic devices via USB.

Framework for Tactile Animation

In this section, we present an animation metaphor that allows users to generate tactile content in the same way as they would create visual animations and play them real-time on a VT array. Figure 3 shows the workflow of this authoring mechanism. Designers create tactile animations on a typical animation tool as shown in Figure 3a. The animation object is placed in space, and the designer adjusts its size on the visual

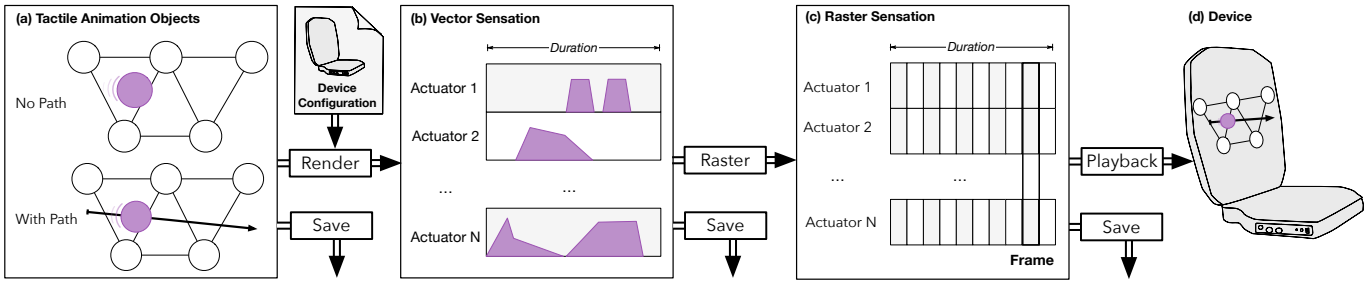


Figure 3: Tactile animation rendering pipeline. Users can: (a) create tactile animation objects; (b) render objects to actuator parameter profiles (such as amplitude) with our rendering algorithm; (c) rasterize vector sensations into frames; (d) play the sensation on the device.

outline of the VT array. The designer then adds movements and special effects to the object using Mango's toolset, and plays it to observe its frame-by-frame sequence.

Mango's rendering engine translates visual animations to tactile animations on the VT array. Knowing the location of vibrating points on the sparse array of VT actuators, the rendering engine resolves the animated sequence into individual actuators using the phenomena of phantom tactile sensations [1, 12]. The phantom sensation is a sensory illusion elicited by stimulating two or more vibratory elements on the skin. Instead of feeling the individual vibration points, the user feels a single sensation in between, whose perceived intensity is defined by the weighted sum of the intensities of the vibrating elements. Therefore, in each frame, the animated tactile object is resolved into intensity of actuators on the VT array (Figure 3b). The rendering engine then calculates raw waveforms for each VT channel (Figure 3c) that can either be sent to the VT device to play the animated sequence or exported as a multichannel datafile for later use. Previous work has interpolated between only two actuators [20, 29]; however, a more generalized 3-actuator interpolation algorithm allows for arbitrary real-time manipulation of the tactile animation object on grid displays.

To accommodate the animation framework, we define three **datatype models**, for use in the current implementation and future expansion of the Mango tool: *Tactile animation objects*, high-level hardware-independent data types for tactile animation; *vector formats*, high-level hardware-specific control common in previous work; and *raster formats*, low-level hardware-specific formats for rendering and playback.

Tactile animation objects are high-level specifications of virtual sensations moving on a 2D VT array (Figure 3a). High-level parameters, such as location, size, and other semantic qualities, can either be constant or variable. Each tactile object has a start time and a duration. Object type is also defined for tactile animations that sets pre-defined parameters and features to animated objects. For example, a moving virtual point can have a position, size, and frequency parameter, while a "rain" effect can have a position and more semantic parameters like raindrop frequency or size.

Tactile animation objects are device-independent. Mango uses a device configuration file (LR4) and the rendering engine to create animated VT patterns on hardware. Animation objects can be combined in novel ways, organized in groups, or generate other tactile animations like a particle generator as in a graphical animation tool, and can have paths that constrain motion to a pre-determined trajectory. We prototyped an early version of the tactile animation object in Mango; however, the data type is extensible.

Vector formats are similar to those in previous work (e.g., [6]). Instead of objected-based definitions, as in tactile animation objects, parameters are defined for individual actuation. (Figure 3b). Parameters include duration, amplitude envelopes (e.g., fade-ins and fade-outs), frequency, and start times. Being device-specific, vector formats offer finer sensation control than tactile animation objects (analogous to pixel-level editing of sprites). However, creating a single percept from independent controls can be challenging. This data type is useful when rendering methods for the hardware are not defined or the user wants to control specific actuator sequence to animate tactile content, such as using the Tactile Brush [12].

Raster format, analogous to a raster-graphics image or WAV file, is suitable for playback operations or exporting it to a device specific format (Figure 3c). A raster format contains a matrix of actuator intensities; each row defines intensities of an actuator and columns containing the intensities at each time instance. Each format also contains a timestamp row defined by the rendering engine's framerate. The playback system parses the raster data, finds the current column, and pushes these actuator settings to the device. This data type is also used for real-time feedback during authoring.

Authoring Interface

The authoring interface allows designers to efficiently create moving tactile content in a familiar environment. Here we describe user interactions, most of which are through the animation window (1) and timeline (2) (Figure 4).

Animation Window: A user creates a tactile animation object (3) with a "new object" button (6), then manipulates it in the animation window (1). The window is overlaid with a faint trace of the VT hardware (13) for context. Here, we used an array of 10 VT actuators (Figure 6).

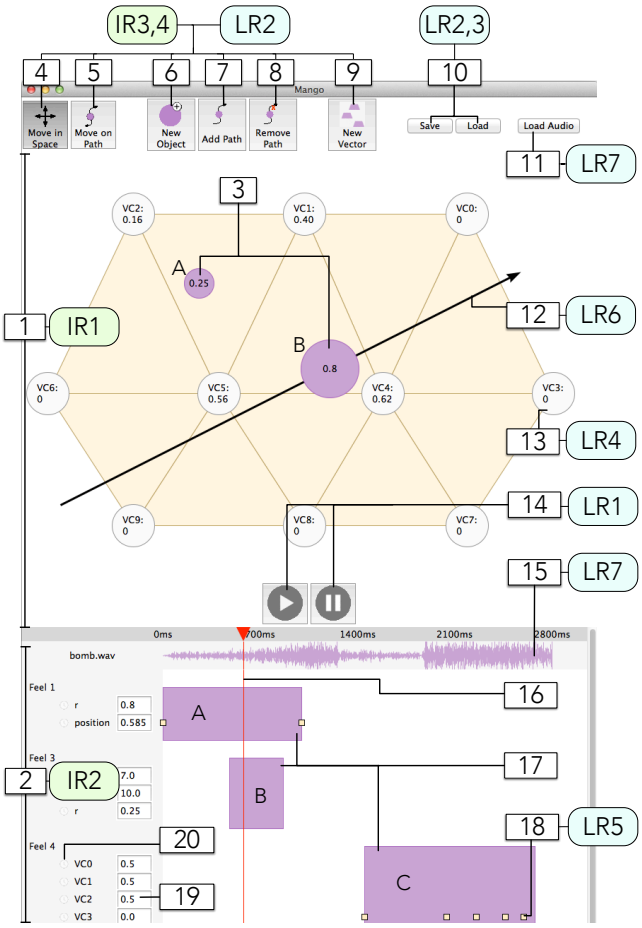


Figure 4: Mango graphical user interface. Key components are labeled and linked to corresponding design requirements.

Object Paths: The animation object (3A) has (x, y) parameters describing position, an "r" (radius) parameter, corresponding to the VT output voltage from 0 (minimum) to 1 (maximum). An optional path can be added to an object (7), or removed (8), along which the motion of the object (3B) is constrained (12). The path-object (3B) is manipulated in two ways: moving on path (5), which moves the object from the beginning (position=0) to the end of the path (position=1), or moving in space (4), which moves the object and the path together on the animation window (1). The current Mango implementation only supports straight-line paths, however their use can be extended in a later version. Also note that curves can be accomplished through keyframed (x, y) positions.

Timeline: Each animation object (3) is represented in the timeline (2) as a track (17). The red scrubhead (16) (shown as a triangle and line) shows and manipulates the current time. Animation objects can be moved in time by clicking and dragging, and resized to change duration. Individual parameters can be set on the left, by typing values into text fields (19), allowing precision. The entire animation can be played and paused using buttons (14) or the spacebar.

Keyframes: Parameters can be toggled as "keyframeable" with a small clock button (20). When the value is changed, a keyframe (18) is automatically created at the current time. Intermediate values are linearly interpolated.

Vector Sensations: A new vector can be created by selecting an object (3) then clicking on a button (9). These sensations control each actuator directly through the parameter values, controlling that actuator's voltage from 0 to 1 (same as the "r" parameter). The corresponding actuator is highlighted in the animation window (1) when the text field (19) or track (17C) is selected. Each track is also keyframeable.

Save and Load: Animations can be saved and loaded (10) to/from JSON files. An audio track can be loaded (11) to the timeline (15).This allows the user to design a VT experience for sound files (LR7). Video overlay is left for future work.

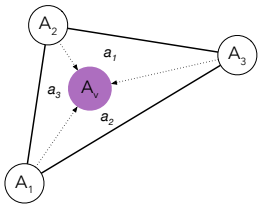
Hardware Configuration File: A hardware-specific structure is defined and stored in a JSON configuration file (LR4). The file contains: (a) physical width and height of the grid, (b) a dictionary of actuator types (e.g., voice coils or rumble motors), each with a list of control parameters (e.g., frequency, intensity) and allowable values; (c) location and type of each actuator; (d) supported communication protocols and rendering methods; (e) brand information (e.g., USB vendor id and product id) for device recognition; and (f) default settings. Physical dimensions are defined in SI units, e.g., meters, Hz.

Playback: Once the animation of the object is defined, the user can play and stop the animation. During playback, the animation runs in (1) and the corresponding parameters vary in (2). Simultaneously, VT stimulations are activated on the hardware for user feedback. Multiple animation objects and vector sensations can exist simultaneously. Actuators output the sum of all the values generated by objects (described later in the Rendering Algorithm section) and vector sensations.

RENDERING ALGORITHM

Mango's rendering algorithm defines how high-resolution haptic feedback is translated to sparse grids of VT actuators. The rendering algorithm translates animations created in the animation window to animated VT patterns on the hardware. Figure 3 shows the rendering pipeline that converts animation objects to a raster format, which outputs to the hardware.

The rendering algorithm is derived from psychophysical understanding of VT illusions on the skin and creates percepts of virtual actuators and their motion in between a set of real actuators. The precise perceptual model depends on several factors, such as type of VT actuators (DC vs. voice coil motors), stimulation site (forearm vs. back) and the spacing of actuators in the array (e.g., [12]). To allow for custom framerates and real-time feedback, we generalize from the 1D case (in between two VT actuator along a line) to the 2D case (in between three or more actuators, previously accomplished with non-VT sensations [34]). Thorough investigation of the psychophysical model is beyond our present scope, however, we empirically determine the most effective model among those documented in the literature for the 1D case with a pairwise comparison.



(a) Barycentric coordinates

Linear $A_i = a_i \times A_v$

Log $A_i = \frac{\log a_i + 1}{\log A_{max} + 1} A_v$

Power $A_i = \sqrt{a_i} \times A_v$

(b) Candidate interpolation methods

Figure 5: Interpolation models used to determine physical actuator output (A_{1-3}) from virtual actuator intensity (A_v) and barycentric coordinates (a_{1-3}).

Perceptual Selection of Interpolation Models

The rendering algorithm translates virtual percepts to a physical actuator grid. We first construct a Delaunay triangulation for all actuators to automatically define a mesh on the hardware grid. At each instant of rendering, we use barycentric coordinates of the virtual animation objects relative to a triangle defined by three real actuators (Figure 5a). Barycentric coordinates are scaled by an interpolation method to determine real actuator intensity.

We propose three interpolation models for Mango, derived from prior psychophysical understanding of phantom VT sensations: (i) *linear*, (ii) *logarithmic ("log")*, and (iii) *Pacinian power ("power")* (Figure 5b).

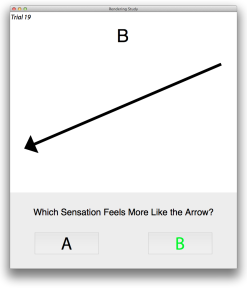
In the linear interpolation model, barycentric coordinates are linearly related to actuation amplitude. In the log model, these coordinates are scaled logarithmically, as perceived intensity is related to physical vibration amplitude [36]. In the power model, coordinates are coupled to the power (square of the amplitude) of vibrating stimulations [36]. Linear and log interpolation models have been used in the past to express either location or intensity respectively (but not both) of virtual sensations between two vibrators [1, 29]. A Pacinian power model was used in [12] to account for both location and intensity of virtual sensation between two vibrators.

Pairwise Comparison Study

To determine the preferred model for this VT hardware in Mango's rendering pipeline, and to identify relevant factors (e.g., frequency, amplitude), we performed a pairwise comparison of our three candidate interpolation models.

Participants and Apparatus

Eighteen volunteers took part (6 female, between age 20-35). The VT hardware consisted of 10 high-quality VT actuators (C2 tactors, Engineering Acoustics, Inc., USA) arranged in a 3-4-3 layout and mounted on the back of a chair in a pad 21 cm high, 29 cm wide, and 2 cm thick; actuators form equilateral triangles with edges of 6.35 cm (Figure 6b). The rendering engine updates at 100 Hz. Through piloting, we determined that the device's on-screen visual outline should mirror the sensations rendered on the physical device. That is, if participants see an animation object on the right side of the screen, they prefer to feel it on the right side of the back. Figure 6a shows the experiment interface, in which an arrow represents the sensation direction.



(a) Rendering study interface

(b) Output device with highlighted actuators

Figure 6: Rendering study setup and user interface.

Methods

We conducted A/B paired comparison tests (two-alternative, forced-choice) to determine the preferred model out of the three candidates. In each trial, participants were presented with two stimuli at a 400 ms interval. Each stimulus is a "straight-line" VT stimulation on the back using one model. Participants were asked to select the stimuli that *best represented straight-line motion* in a variety of directions.

Two durations (500 and 1500 ms), eight cardinal directions, and A/B order were crossed with each model pair, and presented in a random order. For each trial, frequency was randomly selected from 80, 160, 240, and 300 Hz, and intensity from between 10 and 20 dB above detection threshold. Each participant performed 96 trials over ~15min (1728 total).

Results

Each algorithm pair's data was fit to a logistic regression model with participant, frequency, intensity, direction, and duration as factors; direction was grouped into horizontal, vertical, and diagonal. We performed stepwise regression (backwards elimination with $\alpha = 0.05$ and a χ^2 test for removing each factor) to iteratively eliminate factors that were not statistically significant.

Logarithmic vs. Linear. Regression eliminated duration, frequency, intensity, and direction ($p > 0.1$). The resulting model has Nagelkerke $R^2 = 0.135$. Using Bonferroni correction for multiple comparisons, 95% confidence intervals for each participant were computed. 11 participants were more likely to prefer Log over Linear ($p < 0.05$) models; none were likely to prefer the Linear model.

Logarithmic vs. Pacinian power. All 5 factors were eliminated ($p > 0.1$). The overall 95% confidence interval of participants selecting Log over Power was 37.06% to 87.40%, overlapping 50%. We therefore detected no significant difference of preference between Log and Power models.

Pacinian Power vs. Linear. We eliminated intensity, direction and duration ($p > 0.1$), with the fitted model's Nagelkerke $R^2 = 0.0970$. The confidence interval for each participant-frequency combination, via Bonferroni corrections, yielded 22 / 72 participant-frequency combinations selecting Power model over Linear model more than 50% of the time. No one chose the Linear model more than 50% of the time.

Conclusion: Logarithmic interpolation outperformed linear and was equivalent to Pacinian power model. We proceeded with the logarithmic model for Mango's implementation, as the power model did not outperform either of the others.

DESIGN EVALUATION

To evaluate Mango's animation metaphor and expressive capability, we asked media professionals to create a variety of designs. Qualitative evaluation was chosen for rich, focused, early feedback of the animation metaphor and lessons for iteration. A quantitative comparison between tool perspectives is left until more refined tools are developed. We wanted to establish whether this is an effective approach before studying the most effective approach.

Six participants (P1-6, 3 females) were introduced to Mango driving the VT hardware described previously. P1 had experience with haptics but not animation beyond video editing; P2-5 had animation experience but little or no experience with haptics; P6 had no experience with haptics or animation, but was familiar with media tools like Adobe Photoshop. P5 was also involved with the requirement gathering interviews presented earlier. Each entire session took 40 to 60 minutes.

Each participant was introduced to Mango with a training task: designing an alerting sensation using either animation objects or vector sensations (order counterbalanced). Then, each participant was given three design tasks. 1) Primarily *temporal*: create a heartbeat sensation. 2) Primarily *spatial*: tell a driver to turn left. 3) *Context-based*: create a tactile animation to match a sound file. A 3-second sound effect of a bomb falling (with a whistle descending in pitch) then exploding with a boom was chosen, i.e., complex with two semantic components. The wide array of resulting designs can be found in the accompanying video. Mean non-training task time was 5:59 (med 5:38, sd 2:46, range 1:41-13:48).

After each task, participants rated confidence in their design from 1 (Not confident) to 5 (Very confident), primarily to stimulate discussion. All designs were rated 3 or higher; P6 wrote "6" for his sound-based design. The animation object training task was always rated the same or higher than the corresponding vector training task. While suggestive, these ratings were self-reported and from a small sample. We thus did not conduct statistical analysis.

A semi-structured interview followed the design tasks. Participants were asked to compare animation objects with vector sensations, and to walk through the interface to elicit feedback. Interviews were conducted and analyzed by a researcher with training and experience in qualitative research, and followed established methodologies: methods of grounded theory [2] informed by phenomenological protocols [22]. Analysis resulted in four themes.

Theme 1: Animation Metaphor

Participants found the tool easy to use. All six participants were able to accomplish all five tasks (object alert, vector alert, heartbeat, turn left, sound). Participants described the interface as intuitive (P1-5), agreeing that it was an animation tool: *"It's up to the standards of other animation tools"* (P1), *"This is totally animation"* (P2), *"It felt very much like an animation tool"* (P4), *"I'm not an expert when it comes to haptics, but this software seems almost as if it can change the game of designing haptic vibrations"* (P5). Negative feedback focused on polish and feature completeness: *"gotta spline [the keyframe interpolation]"* (P2), *"a couple quirks but there was nothing difficult to overcome"* (P4), *"being able to design your own curve [path] would be really nice"* (P5).

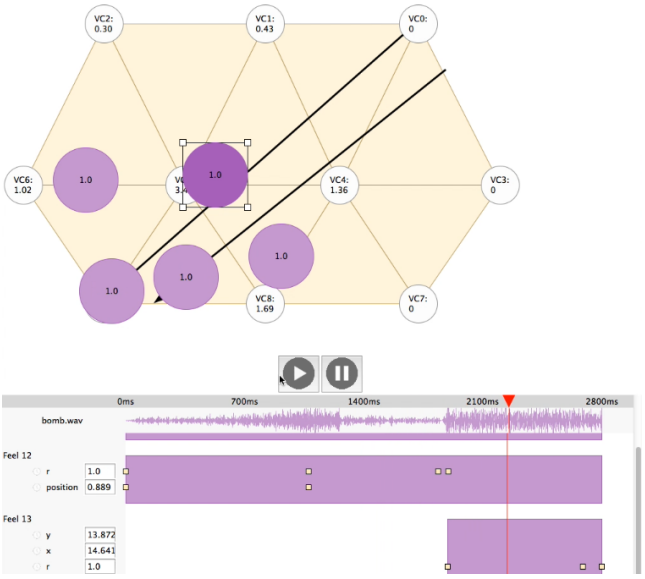


Figure 7: Example of P2's animation for matching a sound. See the accompanying video for all participant animations.

Theme 2: Tactile Animation Object vs. Vector Sensations

Participants relied more on animation objects than vector sensations, which were only used twice: P4's heartbeat task and P5's sound task (combined with an animation object). P1 switched from vectors to animation objects early in her heartbeat task; no other participants used vector sensations.

Animation objects were described as easier to use and more intuitive, especially to represent location or for non-animators. *"After using the new object I'd probably never use new vector again"* (P2), *"easier to find the location of the heart"* (P1), *"if I weren't an animator I think I would only use [animation objects]"* (P4). Vectors were preferred for more fine-tuned control when motion didn't matter as much, often using many keyframes. *"You can control multiple [actuators] at the same time, so you don't have to create new objects and then put them everywhere on the screen"* (P1), *"[Animation objects] can be more comfortable to use when one doesn't work with keyframes"* (P3), *"If you want precise control over [actuators], then vector is the way to go"* (P4).

Theme 3: Designing-in-action with direct manipulation

Participants used direct manipulation to feel their designs in real time, dragging animation objects and scrubbing through the timeline: *"I would make the [animation] object and just play around with it before creating the animation, as a way to pre-visualize what I was going to do"* (P5), *"I kind of play around with it, and randomly come up with the ideas"* (P6).

P2 even noted that YouTube did not have real-time video scrubbing feedback like Mango's: *"I wish I could scrub back and forth [with YouTube]"* (P2). However, continual vibrations were annoying, and participants requested a "mute" feature: *"It would be nice if...it doesn't go off constantly."* (P3).

More generally, participants used feedback from their experience or external examples. P1 stopped to think about her own heartbeat, P2 used a YouTube video of a heartbeat as a reference, and P3 based her alert on her phone: *"It's typical to have two beeps for mobile phones"* (P3). Correspondingly, participants were excited when prompted by an audio sensation: *"I was really happy with the bomb one, because I could really hear it and imagine me watching a TV and then feel it at the same time"* (P1), *"The sound part was good, that would be a fun thing to design for"* (P4).

Theme 4: Replication through Copy and Paste
Replication in both space and time was common while using Mango. Many designs had symmetrical paths to reinforce sensations (Figure 7). All but P4 requested copy / paste as a feature. *"I could just copy/paste the exact same thing on the left side and then move it to the right side"* (P1), *"I have the timing the way I like it, ideally it'd be cool if I was able to copy and paste these, so it would be able to repeat"* (P5).

DISCUSSION

Here we interpret our design evaluation, explore animation with other devices, and describe applications and limitations.

Design Evaluation Summary

From our design evaluation, we conclude that tactile animation is a promising approach for controlling tactile grids. Direct, continuous manipulation of tactile animation objects supported embodied design and exploration by animators, who rapidly iterated on designs to try new ideas. Mango facilitated the design of a wide variety of animations (see accompanying video) and received positive responses. We also found recommendations for our next iteration: more animation features, video as well as audio context, and muting.

Possible Extension to Other Device Classes

The animation metaphor is not limited to a back-based pads. Part of the advantage of an abstracted animation object is that, as long as a suitable rendering algorithm can be developed, the metaphor can apply to other devices. In this section, we illustrate possibilities that we plan to explore in future work.

1D VT Arrays (Figure 8a): 1D VT arrays are common in arm sleeves, wrist bands, belts, and similar wearables. These devices provide sensations along the path of the array. By constraining objects to a linear or circular path, barycentric coordinates collapse into 1D interpolation.

Dense and Sparse VT Grids (Figure 8b): 2D VT grids are also common, used in chairs, gloves, and the backs of vests. While we evaluated Mango with a sparse back-mounted array, tactile animation naturally supports denser arrays, either with our rendering algorithm or by using a nearest-neighbour technique to activate a single actuator.

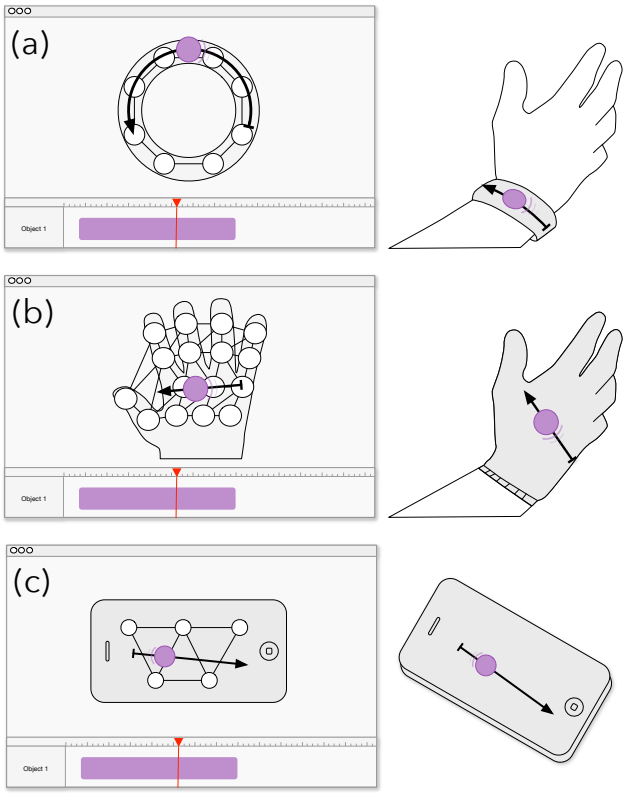


Figure 8: Tactile animation could define motion with (a) 1D actuator arrays, (b) dense and sparse VT grids, (c) handhelds.

Handhelds (Figure 8c): Actuators embedded in handheld objects, such as mobile devices, game controllers, or steering wheels, shake objects instead of directly stimulating the skin. Animators might be able to define source locations for vibrations using handheld-based rendering algorithms (e.g., [29]).

3D Surfaces (Figure 9d): Mango currently only supports a 2D location for its animation objects. However, tactile animation can be extended to support surfaces of 3D surfaces, such as vests or jackets that wrap around the user's body. More work will need to be done to perfect this interaction style, possibly using multiple views or a rotatable 3D model with animation objects constrained to the surface.

Multi-device contexts (Figure 9e): Mango's rendering algorithm already supports connections to multiple devices simultaneously. The editing interface could combine layouts for different devices, enabling animators to animate the entire user experience (such as a car's seat and steering wheel).

Non-vibrotactile devices (Figure 9f): While our rendering algorithm is particular to VT arrays, a tactile animation object can represent manipulable percepts with other actuation technologies. Ultrasound-based mid-air displays generate a sensation as a focal point with a position and size [37]; this sensation could be manipulated through a tool like Mango. Similarly, passive force-feedback sensations (e.g., Hapseat [4]) or height displays (a grid of pins) could be supported.

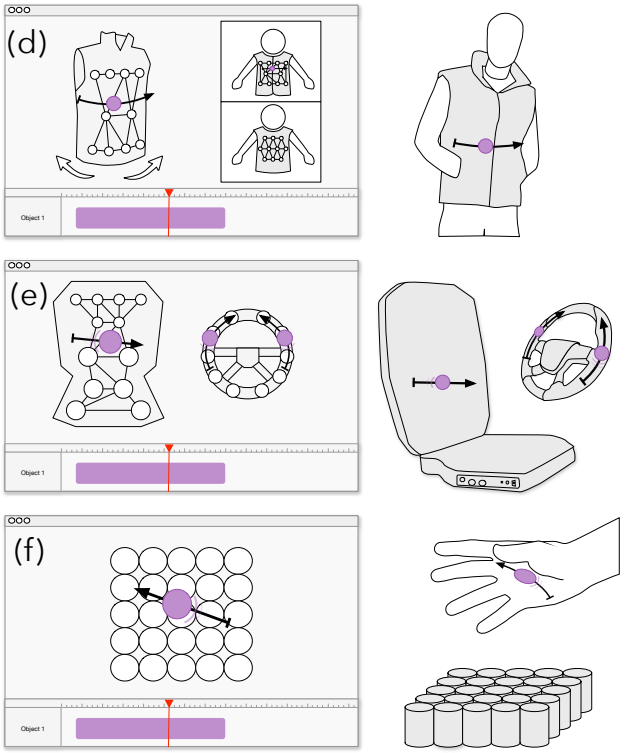


Figure 9: Tactile animation could also define motion with (d) 3D surfaces, (e) multi-device contexts, and (f) non-VT devices like mid-air ultrasound.

Interactive Applications

While our goal was to enable animators to create rich content, the tactile animation object can be linked to alternative input sources for other interactive experiences.

User gestures. User gestures and motion can be tracked and mapped to animation objects directly rendered on the haptic hardware. For example, a user creates patterns on a touch sensitive tablet that maps touch locations to a grid. Users could play games or create personalized haptic messages on the back of a vest. Similarly, a dancer's movements could be tracked through accelerometers, drawing animated haptic content on the body of her audience through actuated theater seats during a live performance.

Camera feed extraction. Motion from video feeds can be automatically extracted with computer vision and rendered on grid displays [16], providing dynamic patterns associated with actions during sports, movies, and games. Similarly, animation parameters could be extracted and mapped to positions on a VT grid, creating haptic feedback for non-haptic media.

Data streams. One main application of haptic grid displays is to provide users directional, assistive, and navigational cues during driving cars, walking down the street, or with oversaturated sensory tasks. Users could associate digital data streams, such as GPS input, to predefined set of directional patterns on the back or palm of the hand.

Limitations

While the tactile animation metaphor seems promising and may apply to many contexts, it is limited by the requirement of a suitable rendering algorithm for target hardware. We have not yet explored other form factors, such as handhelds, multi-device scenarios, or non-vibrotactile sensations. Although we perceptually optimized our algorithm, we did not conduct a full psychophysical investigation. Further work needs to be done to identify the limits, thresholds, and peculiarities of this rendering technique. Examples include: curved trajectories of animation objects (although participants' use of curved motion was encouraging, e.g., P5's turn left sensation), spatial frequency control (how to superpose animation objects of differing frequencies), non-triangular meshes (e.g., quadrilateral interpolation or kernel methods), and mixed actuator types (such as a chair with both voice coil and rumble motors, Figure 9e).

CONCLUSION

This paper introduces the *tactile animation object*, a new abstraction for creating rich and expressive haptic media on grid displays. This animation metaphor allows designers and media artists to directly manipulate phantom vibrotactile sensations continuously in both space and time. Our rendering pipeline, which uses a perceptually-guided phantom sensation algorithm, enables critical real-time feedback for designing. We incorporated these ideas into a prototype, Mango, with a design grounded in animator requirements and haptic design guidelines. Professional animators used our tool to create a variety of designs, giving positive feedback and excitement for future versions. This approach has the potential to accommodate a large variety of haptic hardware, ranging from a single shaking element mounted on the seat to an array of actuators stimulating multiple points on the skin, and can export content into formats applicable in the production pipeline. Tactile animation empowers animators with a new set of artistic tools for rich, multimodal feedback.

ACKNOWLEDGMENTS

We thank our reviewers and participants for their valuable feedback. This work was supported by Disney Research, with additional support provided by NSERC.

REFERENCES

1. D. Alles. 1970. Information Transmission by Phantom Sensations. *IEEE Transactions on Man Machine Systems* 11, 1 (March 1970), 85–91.

2. J. Corbin and A. Strauss. 2008. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (3 ed.). Sage Publications. 379 pages.

3. D. Cuartielles, A. Goransson, T. Olsson, and S. Stenslie. 2012. Developing Visual Editors for High-Resolution Haptic Patterns. In *HAID'12 Posters and Demos*. 42–44.

4. F. Danieau, J. Fleureau, P. Guillotel, N. Mollet, A. Lécuyer, and M. Christie. 2012. HapSeat: producing motion sensation with multiple force-feedback devices embedded in a seat. In *VRST '12*. 69–76.

5. M. Eid, S. Andrews, A. Alamri, and A. El Saddik. 2008. HAMLAT: A HAML-Based Authoring Tool for Haptic Application Development. In *LNCS 5024 - Haptics: Perception, Devices and Scenarios*. Vol. 5024. 857–866.
6. M.J. Enriquez and K.E. MacLean. 2003. The hapticon editor: a tool in support of haptic communication research. In *HAPTICS '03*. 356–362.
7. E. Gunther, G. Davenport, and S. O'Modhrain. 2002. Cutaneous grooves: composing for the sense of touch. In *NIME '02*. 73–79.
8. V. Hayward. 2008. A brief taxonomy of tactile illusions and demonstrations that can be done in a hardware store. *Brain research bulletin* 75, 6 (April 2008), 742–52.
9. S.R. Herring, C.-C. Chang, J. Krantzler, and B.P. Bailey. 2009. Getting inspired! Understanding How and Why Examples are Used in Creative Design Practice. In *CHI '09*. 87–96.
10. K. Hong, J. Lee, and S. Choi. 2013. Demonstration-based vibrotactile pattern authoring. In *TEI '13*. 219–222.
11. W.A. IJsselsteijn. 2003. Presence in the past: What can we learn from media history? In *Being There - Concepts, Effects and Measurements of User Presence in Synthetic Environments*. IOS Press, 17–40.
12. A. Israr and I. Poupyrev. 2011. Tactile brush: drawing on skin with a tactile grid display. In *CHI '11*. 2019–2028.
13. A. Israr, I. Poupyrev, C. Ioffreda, J. Cox, N. Gouveia, H. Bowles, A. Brakis, B. Knight, K. Mitchell, and T. Williams. 2011. Surround Haptics: Sending Shivers Down Your Spine. In *SIGGRAPH Emerging Technologies*.
14. J. Johnson and A. Henderson. 2002. Conceptual models: begin by designing what to design. *Interactions* 9, 1 (Jan. 2002), 25–32.
15. L.A. Jones, M. Nakamura, and B. Lockyer. 2004. Development of a tactile vest. In *HAPTICS '04*. 82–89.
16. M. Kim, S. Lee, and S. Choi. 2014. Saliency-driven real-time video-to-tactile translation. *IEEE Transactions on Haptics* 7, 3 (Jan. 2014), 394–404.
17. Y. Kim, J. Cha, I. Oakley, and J. Ryu. 2009. Exploring Tactile Movies: An Initial Tactile Glove Design and Concept Evaluation. *IEEE Multimedia* PP, 99 (2009), 1.
18. J. Lee and S. Choi. 2012. Evaluation of vibrotactile pattern design using vibrotactile score. In *HAPTICS '12*. 231–238.
19. J. Lee and S. Choi. 2013. Real-time perception-level translation from audio signals to vibrotactile effects. In *CHI '13*. 2567–2576.
20. J. Lee, Y. Kim, and G. Kim. 2012. Funneling and saltation effects for tactile interaction with virtual objects. In *CHI '12*. 3141–3148.
21. C. Moussette and R. Banks. 2011. Designing through making. In *TEI '11*. 279–282.
22. C. Moustakas. 1994. *Phenomenological Research Methods*. Sage Publications.
23. S.A. Paneels, M. Anastassova, and L. Brunet. 2013. TactiPEd: Easy Prototyping of Tactile Patterns. *INTERACT '13* 8118 (2013), 228–245.
24. M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, T. Selker, and M. Eisenberg. 2008. Design principles for tools to support creative thinking. In *NSF Workshop Report on Creativity Support Tools.*
25. J. Ryu and S. Choi. 2008. posVibEditor: Graphical authoring tool of vibrotactile patterns. In *HAVE '08*. 120–125.
26. O.S. Schneider and K.E. MacLean. 2014. Improvising Design with a Haptic Instrument. In *HAPTICS '14*.
27. O.S. Schneider, S. Zhao, and A. Israr. 2015. FeelCraft: User-Crafted Tactile Content. In *LNEE 277: Haptic Interaction*. 253–259.
28. H. Seifi, C. Anthonypillai, and K.E. MacLean. 2014. End-user customization of affective tactile messages: A qualitative examination of tool parameters. In *HAPTICS '14*. 251–256.
29. J. Seo and S. Choi. 2013. Perceptual analysis of vibrotactile flows on a mobile device. *IEEE transactions on haptics* 6, 4 (Jan. 2013), 522–7.
30. R. Sodhi, I. Poupyrev, M. Glisson, and A. Israr. 2013. AIREAL: Interactive Tactile Experiences in Free Air. *ACM Transactions on Graphics* 32, 4 (July 2013).
31. C. Swindells, E. Maksakov, K.E. MacLean, and V. Chung. 2006. The Role of Prototyping Tools for Haptic Behavior Design. In *HAPTICS '06*. 161–168.
32. C. Swindells, S. Pietarinen, and A. Viitanen. 2014. Medium fidelity rapid prototyping of vibrotactile haptic, audio and video effects. In *HAPTICS '14*. 515–521.
33. H. Tan, A. Lim, and R. Traylor. 2009. A psychophysical study of sensory saltation with an open response paradigm. *Proceedings of the ASME Dynamic Systems and Control Division* 69 (2009), 1109–1115. Issue 2.
34. K. Tanie, S. Tachi, K. Komoriya, M. Abe, K. Asaba, and Y. Tomita. 1980. Information Transmission Characteristics of Two-Dimensional Electrocutaneous Phantom Sensation. *Transactions of the Society of Instrument and Control Engineers* 16, 5 (1980), 732–739.
35. D. Tsetserukou, A. Neviarouskaya, H. Prendinger, N. Kawakami, and S. Tachi. 2009. Affective haptics in emotional communication. In *Proc. ACII '09*. 1–6.
36. R.T. Verrillo and G.A. Gescheider. 1992. Perception via the sense of touch. *Tactile aids for the hearing impaired* (1992), 1–36.
37. G. Wilson, T. Carter, S. Subramanian, and S.A. Brewster. 2014. Perception of ultrasonic haptic feedback on the hand. In *CHI '14*. 1133–1142.