Freedom of Movement: Generative Responses to Motion Control

Kate Compton, Michael Mateas
Expressive Intelligence Studio
University of California, Santa Cruz
kecompto@ucsc.edu, mmateas@ucsc.edu

Figure 1: Squishy touchscreen. Rainbow-ified impressions of hands formed by remapping the Kinect's greyscale depth data to a more interesting colorspace

Abstract

Generative methods provide rich, emergent ways to deal with many kinds of data. In this paper, we explore projects that listen to human motion, and respond through emergent generative art in ways that are inspired by dance and puppetry.

Introduction

How do we respond to a body in motion? There are many things in the world that respond to a body in motion, for example, a dancer's physical motion:

- dance costumes, or dance toys like fire poi, physically moved by or attached to the dancer, and subject to forces like drag, momentum, and centripetal force, depending on their materials.
- fields around the performer, as the dancer wades through water or smoke or tall grass, if they disturb curtains as they move
- a human partner, moving their body in response to their perception of their partner's movement. An audience can sense tension, force, and connection, even if the two bodies never touch
- physically unattached collaborators, who, like the human dance partner, "listen" to the movement and respond. This may be musicians or lighting directors who react collaboratively with the movements, or non-human works like interactive projections

All of these phenomena "listen" to the movement of the dancer and respond in some way. As designers of generative systems, we can build systems that operate like any of these real-world responsive systems: our systems can be costumes, fields, physically-connected agents or expressively-connected agents, or to have systems that combine the responsive properties of any of these examples. Observing the range of reactive systems that occur in dance practice reminds us not to limit ourselves to only one kind of "dance partner". In this paper, I reflect on some works where insights from dance (and other movement arts, like puppetry) inform how I can use computers to listen to movement, and respond, collaborate, or amplify that movement.

This paper is emphatically *not* about discrete detection and categorization of gesture. Though we have now spent most of a decade with moderately effective motion-tracking (Kinect, Wii, Leapmotion, Oculus Touch), none of them have sparked the motion-control revolution that each one seemed to promise. In previous work,(Compton and Mateas 2017) I explored how this is driven by a common inability to deal computationally with an input stream that is not a sequence of discretely occurring (and discretely valued) events. There is a broad range of research on performing discrete gesture detection with devices like the Leapmotion(Marin, Dominio, and Zanuttigh 2014)(Potter, Araullo, and Carter 2013), because we can imagine it being used to create the sequence-of-events input that we so often use in our interactive experiences (especially games). However, with a continuous multi-dimensional stream of motion data, discrete techniques like if-statements and categorizations compress the data and lose the continuous fluid quality of the original motion.

Instead, this paper is about how we can use a variety of algorithms (some "artificial intelligence", some not, this paper won't quibble about the definition) to listen and respond to **continuous body movement**.

Listening to Motion

How can we listen to a body in motion? As mentioned earlier, artists have many sensors from which to choose. Some

are relatively basic forms of sensors like accelerometers, gyroscope, distance sensors, and bend sensors. Some sensors operate by processing image data, often via machine learning or various statistical methods, from either a single camera or multiple cameras (often assisted by invisible infrared projections), and several achieve "dead-reckoning" by combining camera, GPS, and accelerometer data.

It is easy to list the ways that we can listen to motion. But let us instead examine *what* motion we listen to, and *why*. Lived human experience informs us that some forms of motion feel better than others. For example, holding arms extended and still is wearying (Nielsen et al. 2003). Yet many Kinect experiences used that pose as a UI technique to simulate a button press. Dynamic loosely-controlled swinging of arms feels *better* than stiff precision, but was underutilized in Kinect games as it couldn't be used to translate traditional UI elements.

One of my first Kinect projectspresented at SF Bay Area 3D Vision and Kinect Hacking, 2/1/2012 took advantage of this. In Kinect Poi, the player used their arms to swing digital fire poi, which left trails of sparks and stars as they swung them. They could then retrace their trails to collect the stars left behind on a previous swing. This had several advantages. The poi were simulated as particles, with continuous acceleration forces, so even when the Kinect sensing momentarily dropped (frequently in old models), the particle continued to move smoothly, without any of the glitching of one-to-one control. Using force-based control, rather than position based control, created a natural "anti-aliasing" effect for the motion input. Finally, the perceived weight of a player's hand increases as they swing their arm, creating the weighted, force-based feedback that was missing from most Kinect experiences. The motion that this art used was the type of motion that felt best for an interactor, and the interaction/"game" was built around that, rather than the other way around.

Lack of haptic feedback or tactile resistance is common in motion control experiences, but this is not unavoidable. In Squishy Touchscreen [1], a user interacts with a soft spandex membrane stretched over a wooden frame. A laser projector backprojects an image onto the membrane, and a Kinect, placed under the projector, maps the deformation of the membrane into a grayscale image. This project was inspired by Kinect musical instruments (like Tim Thompson's Space Palette [2]) where the user waved their hands through the air. Few instruments, with the exception of a theremin, have *no* tactile resistance feedback in this way, so I wanted to create an instrument that you could feel *pushing back*. Spandex acts as a spring and has resistance that increases as you press harder against it. It felt good to press against it, to stroke the screen and feel the drag against your fingers. Additionally, the Kinect could see any deformation of the surface, so the user could press their palm, fingers, face, or any object into the screen, and it would change the character of the deformation.

In another early prototype touch-"screen" (circa 2005), users dragged their fingers through a tabletop full of black sesame seeds (a webcam could see fingertips through the glass bottom of the table). The resistance and physical properties of the seeds provided haptic feedback of resistance, and also produced a very satisfying sound and smell when disturbed. Pressing a finger harder into the tabletop make a bigger "blob" for the image detection to track, and the size of the area of contact could also be felt by the user by the texture contrast between seed and glass.

In Kinect Poi, Squishy Touchscreen, and the Black Sesame Table, the "sensors" themselves are dance partners. Their physical properties (resistance, centrifugal force, inertia, sound, even scent) and the way they encourage interaction (through softness, texture, the pleasure of inertial movement) form a connection with the interactor even before we consider how the digital components of the systems will respond to that input.

Responding to Motion

In Mueller and Isbister's "Movement-Based Game Guidelines", they encourage motion control game designers to not focus intently on game-style interaction: "Start by providing feedback on the movement itself, without too much worrying about scores, multipliers etc. [..] Provide several forms of feedback, but do not require players to engage all of them: better to let players choose which ones to engage based on their cognitive abilities, and shift their attention as mastery grows." (Mueller and Isbister 2014). It can be hard to structure a game with win-conditions (or even resource-logic) around continuous playful motion control, so the fun of these experiences must often come from emergence and surprise rather than control or competition.

Fortunately, one of the major advantages and disadvantages of a thick stream of continuous motion data is that while it cannot be handled by the if-statements of traditional game logic, it does provide an excellent seed for generative methods. Often these methods need not even be complex to be engaging: they merely have to be responsive. The most successful "app" on the Squishy Touchscreen was a rainbow-remapping of the depth field, which I had made as a debug utility. As one pressed harder into the screen, the colors changed around it, like reaching one's hand into a rainbow-colored geode. The stretchiness of the spandex also deformed around whatever was pressed into it, so a hand would become outlined in rings of hand-shaped color. The material was "responding" to the interaction, even before the algorithm got to it.

More complex responses can be designed by passing the continuous motion stream into a pipeline of generative methods[3]. Idle Hands [4] was designed as an installation in an art festival, projected on a wall, that the users control via a Leapmotion. Giant hands (the projection was about 10 feet across) clenched and unclenched even when the controller was idle. When controlled by the user, the hands mostly-

[1](2010, https://vimeo.com/217033311

[2]https://spacepalette.com/

[3]see (Compton and Mateas 2017) for a catalog of the range of generative methods and how they can be used to compose such a pipeline

[4]http://galaxykate.com/apps/idlehands/

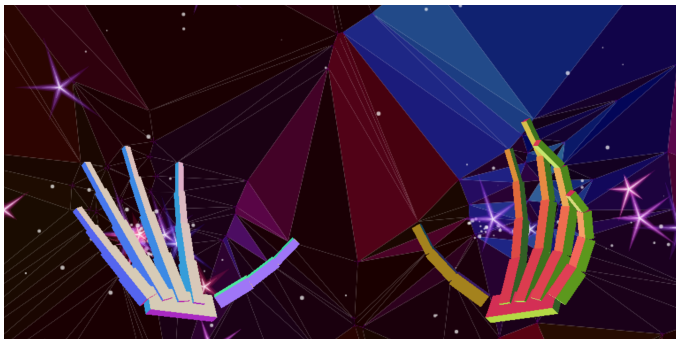

Figure 2: Idle Hands, a Voronoi diagram generated from Leapmotion 3D finger-joint points, with particle system stars

faithfully reflected their hand gestures. The Leapmotion's data stream was a continuous (etd. 40fps) feed of 3D vector positions for all finger joints, which was compressed to 2D points and used to construct a Voronoi diagram of regions and colored as shaded fragments. A few flocks of particles were gravitationally attracted to the fingertips to further accentuate the user's motion. The response to the user data was straightforward, but the directness made the experience rather visceral (many reported a vividly tactile sensation of "crinkling" the background , without touching anything).

One interesting pattern that I discovered with Idle Hands was the importance of flexibility of control. Like the Kinect-controlled poi, any motion control system has moments where tracking drops frames, or the interactor walks away. In these moments, a virtual agent can take over for the interactor. This can be done to patch or smooth the motion, but it can also be used to playfully resist the user's control. Is this a direct mirror, or an intelligent partner mimicking your movements, only to break free with some improvisation? Previous projects (Long et al. 2017) have experimented with the dance partner as an autonomous agent. In my most recent work, I experiment with using the autonomy of the dance agent as a continuous slider.

My most recent motion-reactive art is on dance-reactive puppets.[5] This project was funded by the Google Creative Lab as an experiment to use their Posenet Tensorflow detection algorithm (Oved). This algorithm produces similar skeleton data to the Kinect, only instead of using infrared dots and multiple cameras, it uses machine-learning on normal RGB webcam data, potentially reaching a vastly larger audience than the Kinect ever has. This project was inspired by Nick Cave's Sound Suits (Cave et al. 2010), dance costumes which distort the body into strange shapes and become partners to the dancers, and the Muppets, where the responsive materials of the Muppets (Kermit's flailing arms, Animal's chickenfeathers, Janice's satin hair) become part of their character and movement. The idea was to create generative dance suits whose animation would respond to and exaggerate and reinterpret the movement of a user (as detected through Posenet), just as the physical forms of

[5] http://www.galaxykate.com/apps/puppet/

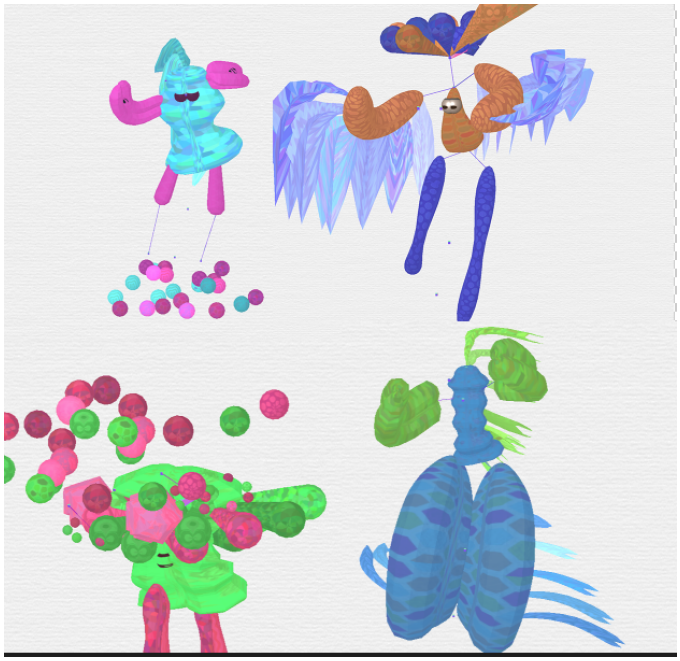

Figure 3: Generative dance puppets with a variety of secondary-motion gesture-enhancing dance accessories When animated, the feathers and orbs emphasize and elaborate on the human user's motion like a dancer's costume

the Sound Suits and the Muppets do with their dancers and puppeteers. I adopted some ideas from the Spore creature creator (Hecker), making the bodies based on tubes, but created more emergent and surprising forms based on the tubes (super-ellipse cross-sections, wrinkles or oscillations along the length of the tube). I also used Spore's wiggles-and-jiggles system of secondary motion (and past work on secondary motion in generative animation (Compton and Mateas 2015)) as inspiration to create a variety of motion-controlled "parts": yoyos, bobbling balloon spheres, fringe, and luxuriantly flowing feathers. Each kind of dance accessory "listens" and "responds", in different ways (to fast acceleration or slow), depending on where it occurs (head or hands or legs), and its physical properties.

At the time of development, I did not have access to the live stream of data from the webcam (that part of the technology was unreleased) so I had to create synthetic data, from a dancing virtual forward-kinematics-animated body in 3.JS, which could generate the data that we anticipated receiving from the machine-learned component. I set up the data-generating virtual body so that it could be driven via a Leapmotion (translating the finger movement into joint movement), the potential future Posenet data, music data, or some combination of all three. It also had a slider that controlled independent noise-controlled data (autonomy) versus user-provided data (mirror mode). One could imagine this slider being driven by anything, including the agent's "boredom" with the player's lack of movement.

The released version of Posenet yields only 2D point data, not the 3D of the Kinect, so I developed a very rudimentary

system to jiggle the 3D synthetic body until it matches the 2D detected points [6] It is far from accurate, yet like much of the work discussed here, it seems that the accurate movement is *far* less important than continuous, reactive, responsive, and emergent movement, and it is an enjoyable "presence" to interact with.

Conclusion

Dance (and movement arts like puppetry) have a long and developed history of turning human movement into something pleasurable, alien, expressive, or transcendent. Movement augmentation both listens to and responds to user movement. Some patterns of listening/responding are **costumes, fields, physically-connected agents or expressively-connected agents**.

Both Squishy Touchscreen and the Black Sesame table were *fields* that the user disturbed with their motion, creating eddies and deformations in the physical interface and also in the digital response. Idle Hands is a field which the user manipulates with their fingers, but while it lacks the physically responsive interface, the seamlessly responsive interaction created an impression of physical touch. The Kinect Poi and the dance puppets are *costumes*: they are linked to the user's movement, but have secondary motion that amplifies and elaborates on that emotion. Like the virtual partner Lumen.AI project, the puppet is an autonomous *agent*, but can move continuously between being a autonomous partner or a costume as its agency is dialed up or down. My projects do not have a strong expressively-connected agent component (I prefer more directly-reactive agent action), but this would be an avenue for exploration for either these projects or any other generative movement-reactive system, such as a musical or visual background improvisation based on some generative interpretation of user movements. These categories only begin to outline the *range* of how interaction in real world dance/movement arts can inspire and inform digital systems; much more exploration in the vast world of dance culture is possible.

References

Cave, N.; Cameron, D.; Eilertsen, K.; and McClusky, P. 2010. *Nick Cave: Meet Me at the Center of the Earth*. Yerba Buena Center for the Arts. Exhibition at the Seattle Art Museum 2011.

Compton, K., and Mateas, M. 2015. A Different Kind of Physics: Interactive evolution of expressive dancers and choreography. In *Computational Creativity in Games Workshop, ICCC*.

Compton, K., and Mateas, M. 2017. A generative framework of generativity. In *Experimental AI in Games Workshop, AIIDE*.

Hecker, C. My Liner Notes for Spore. `http://chrishecker.com/My_liner_notes_for_spore`. Accessed: 2018-08-10.

Long, D.; Jacob, M.; Davis, N.; and Magerko, B. 2017. Designing for socially interactive systems. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, 39–50. ACM.

Marin, G.; Dominio, F.; and Zanuttigh, P. 2014. Hand gesture recognition with leap motion and kinect devices. In *Image Processing (ICIP), 2014 IEEE International Conference on*, 1565–1569. IEEE.

Mueller, F., and Isbister, K. 2014. Movement-based game guidelines. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2191–2200. ACM.

Nielsen, M.; Störring, M.; Moeslund, T. B.; and Granum, E. 2003. A procedure for developing intuitive and ergonomic gesture interfaces for hci. In *International gesture workshop*, 409–420. Springer.

Oved, D. Real-time Human Pose Estimation in the Browser with TensorFlow.js . `https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5` Accessed: 2018-08-10.

Potter, L. E.; Araullo, J.; and Carter, L. 2013. The leap motion controller: a view on sign language. In *Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration*, 175–178. ACM.

[6] `http://www.galaxykate.com/apps/puppet/posematch`

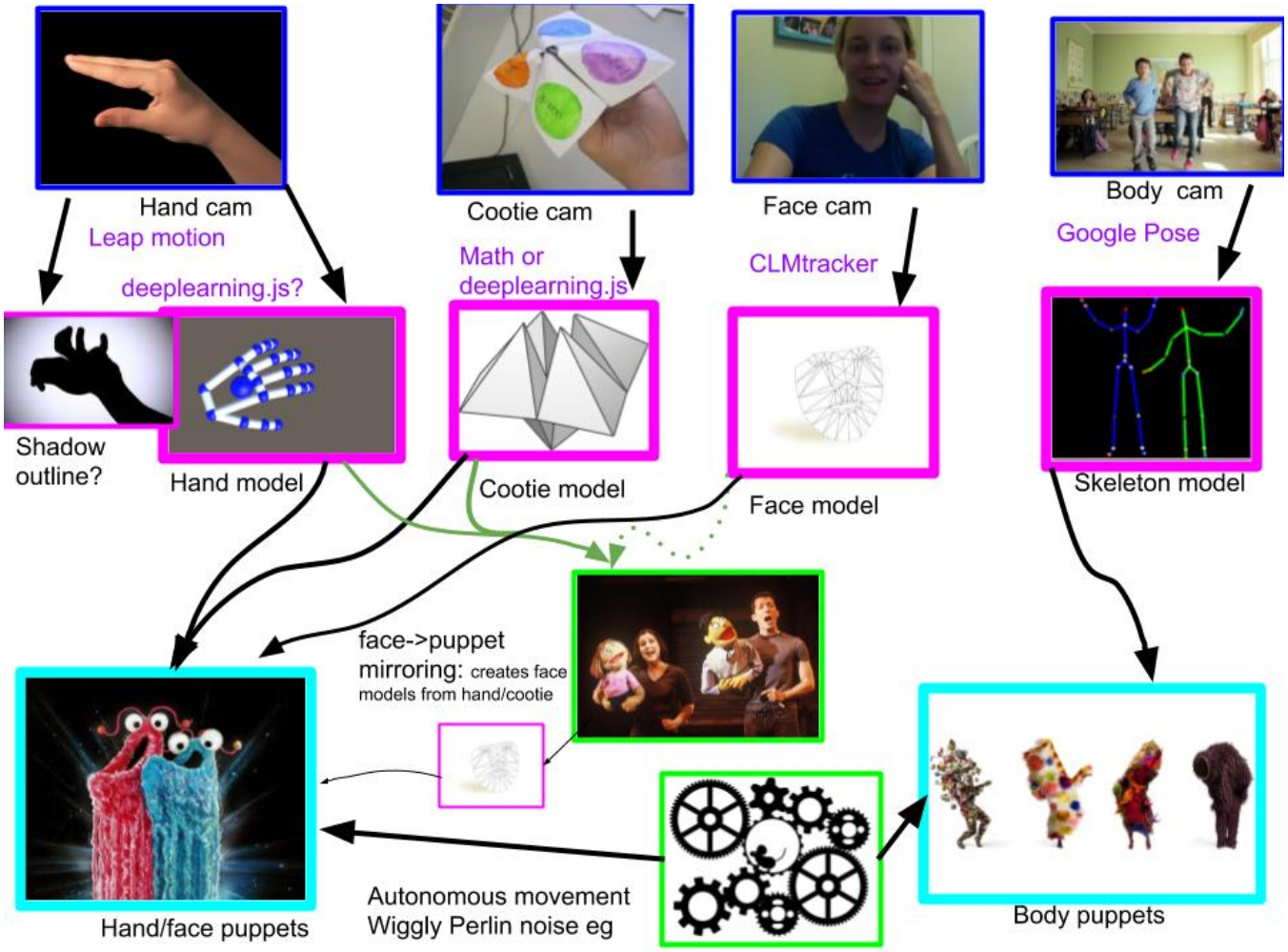

Figure 4: Data flow diagram of the puppet project. Blue outlines are sensors. Pink outlines are processed input streams. Cyan outlines are output graphics. Green outlines are autonomous or puppeted control