Work-in-progress preprint: Live notation for patterns of movement

Alex McLean, Then Try This, UK and Kate Sicchio, Virginia Commonwealth University, USA

Abstract

By exploring how we might create movement through a live notational process of programming in real time, we aim to create a live coding[1] language for algorithmic choreography. Our work spans patterns, computation, movement, notation, robotics and performance. We examine weaving, dance, and musical forms as places of inspiration. We also develop a telematic performance where we live code both the audience and a small robot in order to use practice as a place to further our thinking, tacit knowledge and develop notation of ephemeral processes. We conclude by considering the inexhaustible quality of live, language-based approaches to choreographic notation.

Introduction

Notations are by nature incomplete, often describing continuous expressions in terms of discrete, symbolic operations. How can abstract symbols generalize gesture, with all of its nuanced timings and qualities? This question gets to the heart of how we can notate movement. Choreography can be seen as the organization of gestures in space and time, and yet despite well-known attempts, there is no widely adopted choreographic notation. Acknowledging this, through our collaboration we look for algorithmic choreographies where computer programming is brought closer to dance — where coding and moving are connected in creative feedback. In the following we introduce our approach to the development and performance of dance-specific live coding languages. But first, we mark out some conceptual ground for our work, by challenging some pervasive assumptions which we believe hold the algorithmic arts back.

Establishing dichotomies

Let's start by identifying and clarifying three pairs of concepts that have otherwise become entangled and conflated: *computation* vs *automation*, *determinism* vs *predictability*, and *sequencing vs patterning*. Once we have teased these terms apart, we will then apply them in understanding notation as a live choreographic interface for working with patterns through *non-automated computation*, and *unpredictable determinism*.

Computation vs automation — These two terms are too often needlessly conflated, but here we are mainly interested in considering computation *without* automation. Once we separate these two terms, we open up all the possibilities of human action-as-computation, including its long, partially hidden history in traditional craft culture. After all, humans have always computed things, as is clearly evident in how handcrafts such as weaving and braiding are full of algorithms for generating patterns. Craftspeople have explored computation well before automation, and while the Jacquard device is

[1] Live coding is a practice largely situated within the performing arts, where a performer creates a live work by writing code while it is being executed.

often cited in connecting weaving with computing, handweaving has always been computational. Jacquard's device only stands for automating the computation, thereby taking it out of human hands (Harlizius-Klück 2017).

As well as in craft, human action-as-computation is also seen in dance and choreography, wherever rule-based systems are used to create and perform movements of the body. A mark of this is where simple instructions are used to generate complex choreographic movements and relationships. One such computational approach can be found within Trisha Brown's choreographic work *Locus*, which does not involve electronic computers, but does involve a distinct system for performing movement, through a diagrammatic cube that assigns letters in space around a dancer. A text (Brown's biography) is used to determine the positions of the points in space. In *Locus*, the focus is on spatial placement of gestures and not the movement itself, resulting in different sections of the piece having different movements (Sulzman 1978).

Algorithmic systems for organizing movement are common in social dance and folk traditions too, such as European maypole dances and the comparable but more complex Tamil dance of Pinnal Kolattam. In both cases each dancer is given instructions to follow, dancing in and out in opposing directions to create a patterned braid around the maypole, or in the case of Pinnal Kolattam, in three-dimensional space. The more involved the dance, the more complex the resulting braid, potentially creating surprising interference patterns where different colors of the ribbons interact according to the particular way that the structure is braided.

Over the past few decades, such systems-based approaches have developed in stage dance too, where they have become collectively known as *algorithmic choreography*. This term is often associated with Merce Cunningham and his use of the *Dance Forms* software in the 1990s (Schiphorst 1993), but this is a late case that focuses on automation of the choreographic process through the use of computers, rather than working with computation as a material in its own right. Choreographer Jeanne Beaman's earlier work from the 1960s also aimed to automate the process of the choreographer, creating dances from chance procedures (i.e. rolling a dice) by using a computer to select movement, duration and space from a list. The computer created "70 dances in 4 minutes" (Eacho 2021). However again, these examples tend to view the computer as a sequencer and randomiser, automatically selecting movements, and putting them together in new combinations, but not really *computing* anything. By contrast, the maypole or Pinnal Kolattam does compute something tangible — the interference patterns emerging from human movements being captured as potentially complex braids.

Computation in dance can be human or technology-based, bringing together processes that follow rule-based systems to create choreography. These algorithms may be simple or complex, designing movement through gestures, spatial patterns or timing structures. Automation, like in the case of machine weaving, may use computation to create choreography that is then executed by movers, but we believe that automation needs computation, more than computation needs automation.

Determinism vs predictability — A system can be both deterministic and unpredictable, in that the results of running a process can be practically impossible to predict in advance, yet when you run the process a second time, it produces the exact same results. Theoretically speaking, we know this from the halting problem in computer science; there is no general procedure for telling whether a given deterministic procedure will complete. That is, we can't always predict whether a given deterministic procedure will return an answer at all. We also know deterministic, unpredictable systems from chaos theory, in that if we make a very slight change to a deterministic, chaotic system, it will give completely different results.

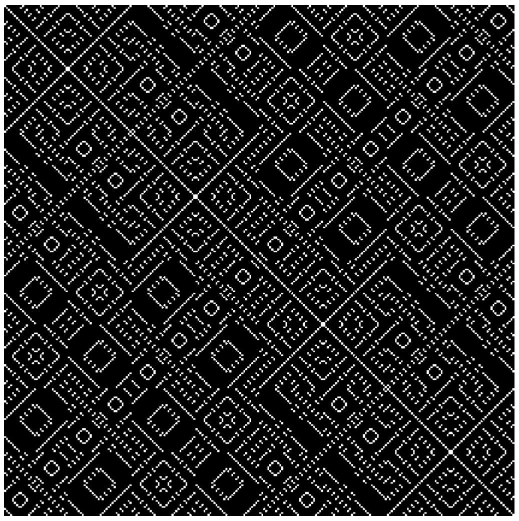
This difference between determinism and predictability can be seen in the way that random number generators are often used in media arts to create variation in results. These are known as *pseudo*-random number generators, because they are deterministic, only giving variation by including time and state in their calculation. They are designed to be unpredictable through the use of chaotic interference patterns, creating deterministic sequences that exhibit the properties of noise. So they are a good example of deterministic unpredictability. However, pseudo-random number generators are not what we are interested in here, rather we are addressing the creative possibilities of composing unpredictable, deterministic patterns by hand, through non-automated computation. This comes not from work with 'black box' random number generators, which make arbitrary choices. Rather, we see creative possibilities in hands-on experimentation with algorithmic pattern operations, for example repetitions, symmetries, and interference patterns at multiple scales, to create unpredictable, complex results from simple parts.

Sequencing vs patterning — The third and final dichotomy that we'd like to draw is to separate linear sequences from patterns, despite the latter generally arising from the former. The need here is to clarify what we really mean by *pattern*. This word is applied across many fields including mathematics, arts, crafts and design, but carries a different weight and meaning in each of them, and in some cases is even a pejorative term. For instance in music, *pattern* tends to stand for any repeating, discrete sequence, and therefore patterns tend to be loved by electronic dance musicians, and derided by 'serious' classical composers (Hugill 2020). On the other hand in textiles, the word *pattern* may refer to a technical procedure for a traditional craftsperson to follow, such as numerical braiding patterns indicating the relational exchange of threads, or procedural knitting patterns where the craftsperson follows instructions to loop, jump and switch between parts of the pattern in order to produce a garment of the correct size. Here then, we refer to *sequences* as linear or step-by-step, and *patterns* as more nuanced, branching and recursive procedures where symbolic elements may be repeated, but are also transformed at different scales, and aligned and composed together to create complex results.

This dichotomy of sequencing vs pattern follows from the previous two dichotomies we drew between computation vs automation, and determinism vs predictability. Automatism tends towards simple, predictable step-by-step sequences, not only to make procedures easy to automate, but also to make procedural breakdowns easier to recover from, when a human operator has to step in (Bainbridge 1983). Computation on the other hand embraces unpredictability in combining potentially simple elements in order to produce surprising, complex forms, as we will explain next.

Complexity from simplicity

If we embrace computation and determinism, and reject automation and predictability, what are we left to work with, in terms of creative material? Our answer is an approach that focuses on perceiving the complex and unpredictable results of simple pattern-making processes. We use the word *complex* with care, however. Where artists engage with computation, there may be an assumption that they are doing complex, unfathomable things. We argue that this would be a failure to grasp the affordance of computation, as a means to generate and experience complex forms from simple structures. To support this, we offer two examples; a contemporary bitfield pattern, and a heritage weaving pattern, shown side-by-side in Figure 1.


a) Bitfield pattern, by Martin Kleppe

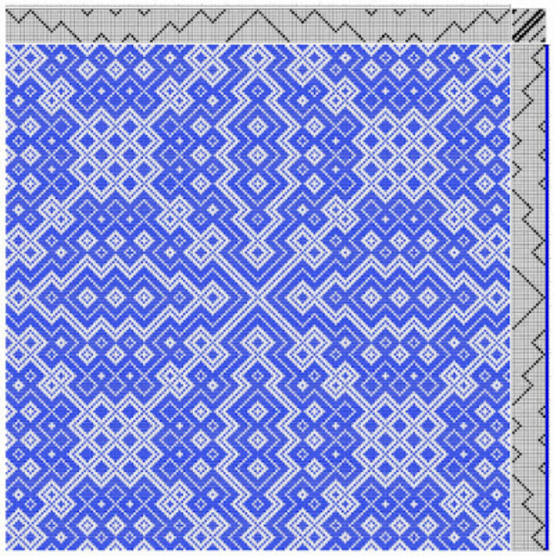

b) Traditional "Earl's Canvas" weaving pattern, rendered by Denise Kovnat

Figure 1: Examples of two-dimensional algorithmic patterns

Figure 1a) shows a 'bitfield' rendering of the function `(x ^ y) % 9 == 0` for each `(x,y)` point in two-dimensional space.[2] The ^ operator stands for a bitwise exclusive-or, and % for modulo. By aligning the bits of two 'x' and 'y' numbers, combining them with this boolean operation, and painting the respective pixel white where the result divides equally by 9, an interference pattern is revealed. Figure 1b) shows a similar interference pattern, but from handweaving.[3] Interference is again set between x and y directions (which weavers refer to as weft and warp respectively). This pattern arises from a binary matrix multiplication of the binary values from the shaft loom configuration shown at the top, the feet movements (known as treddling) shown on the right, and the connection between them (known as the 'tie up') in the top right. The resulting weaving pattern (shown centrally in blue) shares some visual similarities with that of the bitfield pattern on the left, despite the manner of construction being very different. However, we should give proper respect to the far older and more developed craft of weaving. There is much more that we could vary in the weaving block design, for example by introducing a pattern to the selection of color across the warp and weft threads. We should also bear in mind that this two-dimensional design represents an entanglement of threads into three dimensions, a process which creates visual and physical properties far more complex than what is represented in two dimensions on the screen. But still, the family resemblance between these computational crafts, one contemporary and one ancient, is striking.

[2] For community discussion of bitfields and related practices of bytebeat and sizecoding, see https://forum.algorithmicpattern.org/t/bytebeat-tixyland-and-other-functions-of-time/396

[3] For discussion of this pattern, and its translation to echo threading, see https://www.denisekovnat.com/2020/11/gebrochene-echo-and-jin-with-fiberworks.html

Although perhaps simple, these coded patterns provide very rich ground to explore. This is the affordance of language — instructions as parts that can be mixed together and rearranged in a multitude of ways, as interoperable fragments that may be abstracted or recursively embedded, creating a generative, combinatorial explosion of possibility. Its unpredictability means that every change might feel like a step into the dark, but where the results are immediately generated, as in the practice of live coding, each change can instead feel like switching on a light.

Having asserted our interest in computation but *not* automation, and deterministic procedures but *not* predictability, we are better equipped to seek notations for algorithms that are for humans to follow, through tacit, embodied engagement with movement through computation.

Histories of notating movement

Before introducing our own recent work in notating movement, we will first review related historical and contemporary approaches, allowing us to reflect upon and bring together the different histories of code and choreography. Let's first consider the representation of human exertion as *effort*. A key reference point in the notation of effort is in the book *Effort: Economy in Body Movement* (Laban and Lawrence 1947), which proposes a systematic, seemingly elegant way to categorize human efforts in movement, through combination of elements such as time, weight, and space. This work was written by choreographer Rudolf Laban in collaboration with management consultant F. C. Lawrence in 1947, and as such is as applicable to time-and-motion studies in industry as it is to choreography in dance. We should also note that Laban's earlier work was as choreographer to Hitler and Goebbels, until his work on the 1936 Olympic games was rejected (the Nazis rejecting Laban, rather than vice-versa). It is interesting that while this theory arose from a background of white supremacy and Fordism, it continues to be influential in choreographic research and practice.

In South Indian Carnatic dance styles, spoken syllables are used to represent different movements, strung together to form movement phrases or *Jatis*. These syllables may be written as notation to aid memory, but the emphasis is very much on orality, with different, overlapping sets of syllables associated with different dance styles (Seth 2017). This approach extends also to Carnatic music, where syllables are central to the conception of rhythm, applied both to represent drumming articulations (e.g. on the mridangam drum) or directly in vocal performance, a practice known as *Konnakol*. These ancient, yet still actively developing dance traditions pose a challenge for our otherwise Western-centric view of notation; a set of non-lexical syllables are used as symbols that represent gestures, but in spoken rather than written form. Because our interest is in *live* notation, where a written notation is transient in that it changes along with the piece it represents, this challenge is central to our ongoing work.

One example of a live notation system in stage choreography comes from Michael Klein and Nick Rothwell from the late 1990s. Their system Choreo/Graph provides a live score that is visible to dancers on monitor screens at the sides of the stage. This system allows the dance to be changed in real-time while being performed, most notably in the work Duplex (deLahunta 2002). Here a cueing system was created that gave dancers either sequences of choreographic material or tasks to manipulate the movement. The liveness of this notational score demonstrates a digital system for dance that changes during the performance rather than a fully set piece created via algorithmic processes.

Live notation was also the name of a research network bringing live coders together with live artists[4]. Both of us (the present authors) took part; the network was convened by Alex in collaboration with live artist Hester Reeve (McLean and Reeve 2012), with Kate a core member able to bridge both disciplines through performance practice. A far-reaching outcome of this exchange between practices came from the recognition of a shared approach which project member Emma Cocker (2014) identified as *kairotic coding*. She provided a philosophical reflection on the role of notation in live improvisation, using the mythological figure of Penelope's *un*weaving as metaphor for a notation that can be unraveled and rewritten. This prefigured later collaborative work with weaver and mathematician Ellen Harlizius-Klück, which took the connection between live coding and weaving further, beyond metaphor and into direct correspondence (Cocker 2017). Indeed, we have already discussed the close correspondences between weaving and coding in the previous section.

With these examples in mind, drawing on concepts from dance notation around documenting gesture, representing the ephemeral as in Konnakol, and live notation that changes live during performance such as with Choreo/Graph, we began to explore how to create a programming language for live coding movement of humans and robot performers. On this basis, we are interested in using computers for the live generation of choreography, focused on the computational and not automating processes. We do this through the approach of live coding, which allows for continuous decision making in both generating and responding to algorithmic patterns. By bringing robots into the work as performers, we look to find connections between humans and machines through language and movement, bound together in liveness.

Live coding robotic performance

We introduce our own recent work in algorithmic choreography, as collaborators working at a distance between Richmond US (Kate) and Sheffield UK (Alex), building upon our in-person collaborations while previously living in the same city (Sicchio and McLean 2017). At such a distance, it was perhaps natural to explore choreographic notation of robotic movement, and we did so using off-the-shelf ROBOTIS components (AX12A servo motors, controlled via an Arduino interface).[5] We worked with two identical robots, one in each studio, with each robot having three servo motors, thereby providing three 'degrees of freedom' (see Fig. 2). Despite Kate's prior and ongoing work with choreographing robots (Sicchio et al. 2022) and our earlier collaborations, we decided to come to this work afresh, re-approaching algorithmic choreography by thinking through practice.

[4] Live Art is a fine art/performance art tradition, using live bodily action as medium.
[5] For technical information about working with ROBOTIS servo motors, see the following blog post: https://slab.org/2022/02/28/making-robots-with-ax-12a/

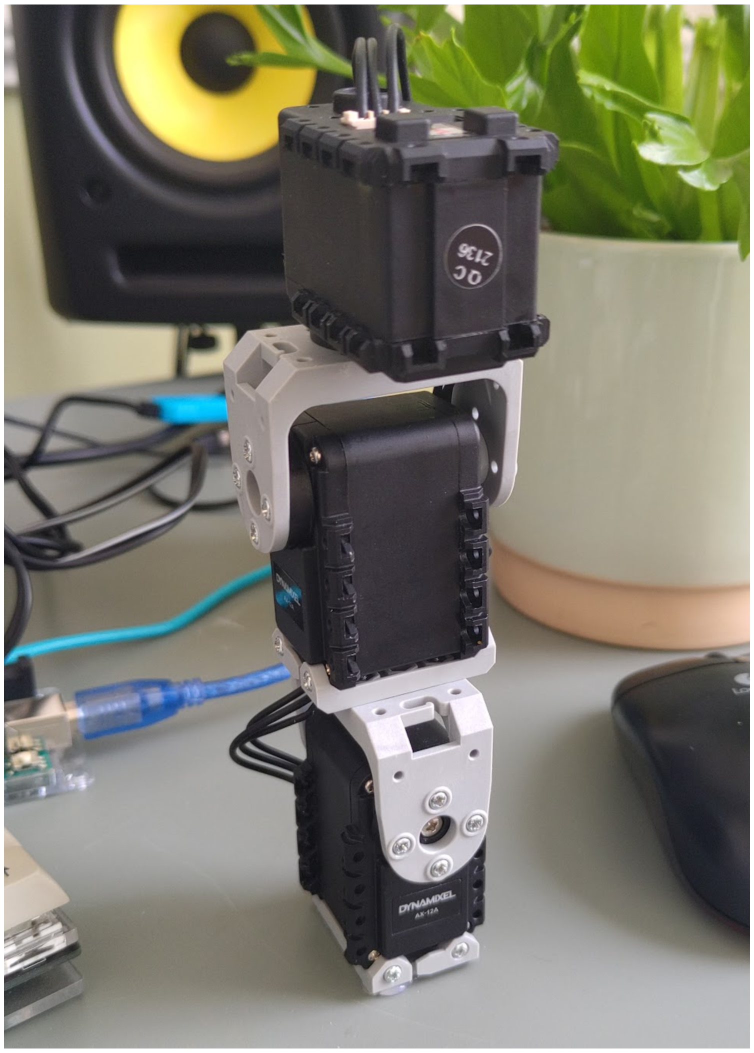


Figure 2: One of our robots, created from three ROBOTIS servo motors coupled with standard fixings, creating three degrees of freedom.

We began by triggering movements of our robots using the Strudel live coding environment for algorithmic pattern (Roos and McLean 2023), which is a port of the TidalCycles live coding system to the web. Despite being in different continents, we were able to work in the same editor via the *Flok* live coding environment[6], which is designed for network music. Flok allows multiple people to code simultaneously within the same editor, where each participant has their own 'cursor'. When someone triggers a live edit, the running code is automatically updated on all participating computers. Despite being designed for making music, the combination of the Strudel javascript environment and Flok editor worked well for patterning servo motor activity, rather than sound.

This initial configuration therefore created the following chain of action, each link having a different bearing on the result:

Kate and Alex → Flok editor → Strudel language → Arduino microcontroller → Servo motors

[6] The free/open source Flok editor for networked multi-user live coding is available at https://flok.clic.cf/

Being a live coding editor, the Flok system allows the creation and manipulation of code, while it runs (Blackwell et. al. 2022). Because the development of that code forms the structure of the piece, live coding performances tend towards progressive building up and reduction of structural complexity, without sudden changes or shifts of mood. Although largely designed for making music, the Strudel language is essentially a system for combining sequences and transformations, at multiple scales, in order to generate algorithmic patterns. Its use therefore tends towards generating complex patterns from simple parts, with fractured symmetries that at first may seem arbitrary, but through repetition an underlying structure is revealed. So already, the live coding editor and language environment provides affordances for us to explore.

Ceding control

We could stop here, at the notational level of the live coding environment, in framing our piece. To do so would be to focus on the notation of movement, rather than the movement itself, and its influence on the human choreographers. Indeed, this is how we initially approached control of our robots, looking for ways to pinpoint a position in space using the notation, instructing each robot to move to that position, through a technique called *inverse kinematics*. This extreme level of control is made possible with robots, in that what you write on the computer, can be what you get.

On closer examination though, such an approach to notation as a technology of control[7] became a losing battle. Firstly, in our case, it is not possible for a given extremity of a robot (such as the end of its 'nose') to move to any given point — each robot only has three motors, configured as a mechanical waist, back and head respectively, and because each base is fixed to a table such a robot can only contort to touch certain ranges of positions within their reach. Even if the robot had freedom to reach any point that we might want it to, we would still have to carefully consider its physical characteristics. It takes time to move, and so to reach a particular point at a particular time, you have to start early. If you move too fast, the robot will overshoot, and resonate back and forward until it settles to the intended position. With servo-motors, fast movements are also extremely noisy. We found the more we attempted to exert control over our robot through precise instruction, the more such uncontrollable, unwanted elements had their bearing on the work. So we instead worked with the robotic quirks as creative material, exploring these issues of vibration and timing as resonances and rhythms that are intrinsic to the robot, offering up material affordances that we chose to embrace rather than work against.

In order to work with any robot then, we first need to understand its physical constraints, as creative material to work with. Indeed, this is true also of working with humans. For example percussionist Jaki Liebziet, known as founding member of the experimental band Can, developed an approach to drumming with a notation based on the binary dot-dash familiar to morse code (Podmore, 2020). The constraints of this system were based on the observation that in order to perform a louder 'accented' strike of the drum, the hand needs to move the drumstick further, up and then down, which takes additional time. Liebziet's system therefore worked on interplay between left and right hands, where due to the laws of physics, an accent with one hand requires time provided by a preceding double movement with the other hand. From these simple rules, complex rhythms and paradiddles are generated. Accordingly, rather than seeing the physical constraints of a human or robotic agent as

[7] For an exploration of technologies of control, versus technologies of work, see Ursula Franklin's transcribed lectures "Real World of Technology" (Franklin 1999).

getting in the way of an ideal notated piece, we instead take physical constraints as a starting point, therefore providing a rich creative space in which to work. In that spirit, we turn now to focus on the physicality of our robots.

The Robot

As mentioned earlier, we each have one robot, sitting on our respective desks in Richmond and Sheffield, made from off-the-shelf servo motors. These motors not only allow external control over where they travel, how fast, and with how much power, but also allow sensing of their current position, and pressure applied to it. These physical properties of the robot present a number of constraints. For example, it isn't possible to increase the speed of the robots without increasing the noise that it makes. The robot also isn't able to move around or through the surface it is mounted to. More subtly, care should also be taken to avoid overusing certain movements that wear out the components, by overheating them, or over-stressing interconnecting wires.

Such physical properties offer us a number of choices in deciding how we interact with the robot, for example:

a) If the robot is already moving to a position when you tell it to move to a new one, should it complete its current move, or immediately adjust its course?
b) Should the robot simply move as quickly as possible to the new position, or should its speed follow a particular curve of acceleration and deceleration as it reaches its goal?
c) Should a movement be expressed in terms of the average speed of movement, or in terms of the time the movement should take?
d) Relatedly, should a movement be timed in terms of when the robot arrives at its target position, or in terms of when the movement begins?
e) How about relative movements, like 'switch sides', or 'move halfway to the ground' — should these movements be relative to the current target position of the moving robot, or to the current position?

To a large extent, all of the above constraints and choices apply just as well to humans as to robotics, with the exception that human muscles tend to move more quietly than servo motors — but human movements have their noises too, such as gasps, squeaks, and clicks. The primary difference then is that with robot performers, answers to such questions tend towards control, and with human ones, the answers tend towards respecting human agency. In considering them together, we are therefore able to make these choices explicit, and open to interrogation.

As live coding language designers, we work by defining ways of making things, via the creation of a notation. This means that once we identify choices such as those above, we are able to defer those choices, by turning them into an option, codified as a parameter to a computer language function. This allows us to not only make these choices during a live performance, but change our minds, or even write code to vary the choice over time, following its own pattern.

In our case, we took the choices that most respected the characteristics of the robots we were working with, with the goal of finding movements as interesting choreographic material, rather than direct instructions. This relates to what Ingold (2011) criticizes as *hylomorphism*, where a maker forces their ideas on material, rather than working *with* material through creative feedback. For example, considering choice e) above, if we instruct a robot to 'switch sides', it should move a servo until it mirrors its starting position; but what if we quickly tell it to switch sides again, while it is still moving? If we respect the imaginary world of the notation, the robot should go back to where it started, but if we respect the physicality of the robot, the robot should switch sides relative to its current position. In the latter case, if we keep interrupting its movements with the same 'switch sides' instruction, each movement will be of a shorter distance until the robot settles at its center, orthogonal to the ground. By respecting and perhaps embodying the robot in this way, we have discovered a new gesture afforded by it, a movement that resonates and settles.

From sequencing to pattern — introducing the robot to the audience

The syntax of our language consisted of parts of the robot or body, predetermined gestures, and spatial-temporal relations. Our robots consist of three motors, which we earlier loosely referred to as “head”, “waist” and “back”. Through initial explorations we created a small collection of simple gestures including “move”, “lean”, “switch”, “toggle” and “reset”. Other, higher order movements such as “sway”, “wiggle” and “diagonaltwist” were added later, described in terms of the composition of basic signals (e.g. sine, sawtooth and square waves, and white or perlin noise) directed to the different motors. The distance by which the motor would move, and the duration of the movement were also programmable for each gesture. A simple program to move the top motor of the robot to 25% of its range over a quarter of a second would look something like this:

```
part("head")
.action("move")
.to(0.25)
.dur(250)
.robot()
```

This code could be looped, and new gestures, timings and spatial commands could be added, creating compound movements with unique and often surprising characters. It was then quick to develop sequences of evolving gestures through combining movements. By harnessing the expressivity of Strudel we were not only able to sequence the movements of the robot, but start to create more complex patterns through transformation and combination. This allows a series of gestures to be performed and combined at different scales, allowing patterning of choice or probability.

As part of this work we presented a live coded choreographic performance at The International Conference on Live Interfaces in June 2022, organized by conference chair Adriana Sá and CICANT at Universidade Lusófona in Lisbon, Portugal[8]. During the performance Alex was in Sheffield UK and Kate in Chicago US, both using the afore-mentioned networked Flok editor to program movement and sound, while using video conferencing to engage with the Lisbon audience.

In this telematic performance we introduced our live notation techniques first as a score for audience participation, only introducing our robots later in the piece. The piece began with us sharing our screen with the audience in Lisbon, projected onto the back wall of the largely empty stage of the concert hall. A camera was also set on the stage for us to view how the audience was moving, allowing us to live code their gestures. Different actions were highlighted, with the instructions patterned by our code read out loud by a text-to-speech synthesizer. These instructions were simple to begin with, for

[8] Video documentation https://youtu.be/Lh8yAOT3WOM?t=2810

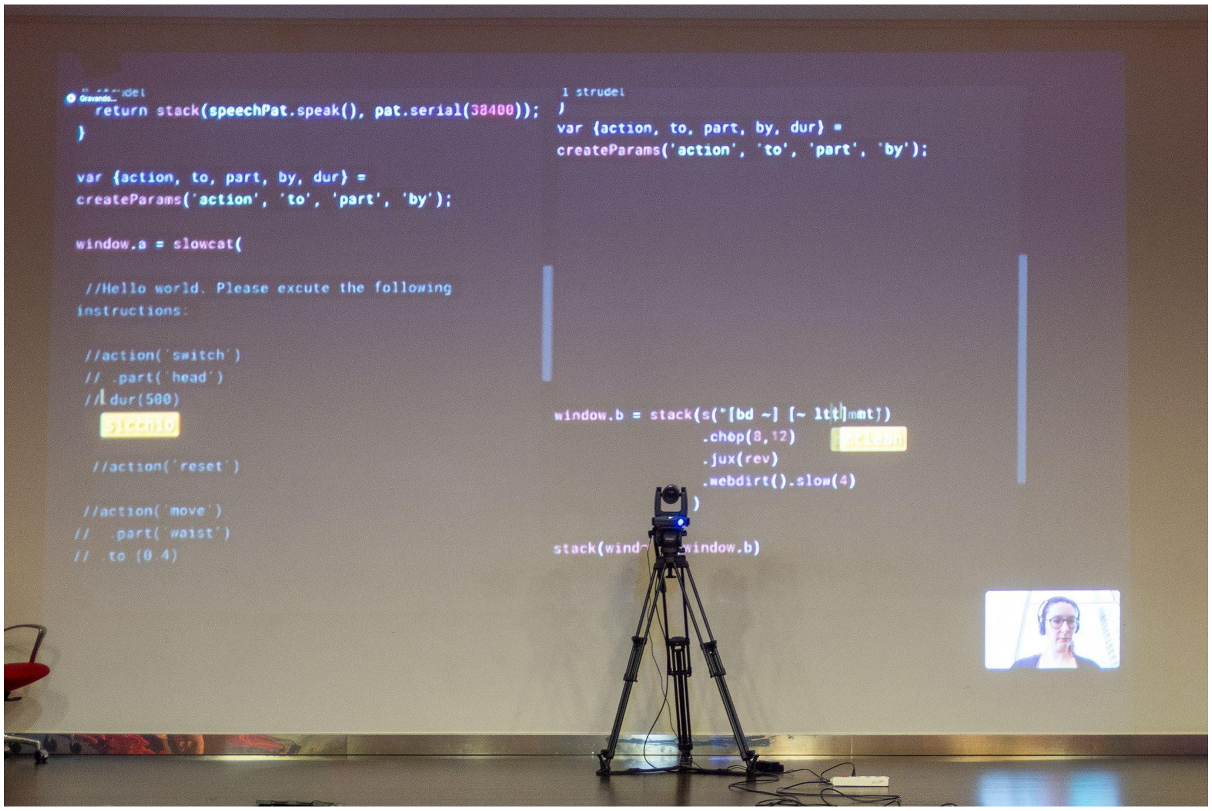
example "move your head thirty-five percent, over five seconds". To accompany the choreography and audio directions, minimal, patterned musical rhythms were live coded by Alex.

We asked the audience to participate in performing these patterned movements, instructed via the code. To facilitate this, the performance began with simple instructions, building up into a structured sequence. But as the piece progressed, more complex movements were created, by composing the simple sequences together and transforming them with multiple patterning functions. The audience attempted to learn and perform the movements in real-time, as they grew in complexity. As the audience began to struggle to keep up with the live coded choreographic instructions, we revealed an additional window on the screen. In this square the robot was revealed to the audience for the first time, shown following the same live coded instructions. In its particular way, the robot was able to continue to perform the actions as instructed, through their increasing complexity.[9]

The interpretation of such code feels very different from human to human, compared to from human to robot. Some people were sitting and some were standing. Some people did a gesture once and paused, while some looped the movement over and over again. Some became single-minded in completing the instructions even when they became more and more complex and difficult to perform without time to break instructions down into smaller pieces. Most humans gave up, sat back and watched as the piece progressed. However, the robot was able to take on each code change without challenge. Its dance was not limited to its ability to keep up with the algorithmic changes in the system, as the work progressed from sequences to patterns. While the notation remained relatively simple, it generated a complexity that only machine movement could execute live.

But still, the human was still 'in the loop' as choreographer of the robotic movements, responding to them by live coding. Within the narrative structure of the piece, in particular the building up of pattern, and the reveal of the robot joining the audience in executing the code, all the instructions for patterning movement were improvised by us as live choreographer-programmers. By placing ourselves as humans improvising with computational movement, rather than automating movement as a prepared script, we created and manipulated patterns throughout the piece, responding to each subjective moment in time. While the robot may highlight differences between human and machine performers, the live coding process demonstrates the human labor of notation, and the ways in which we can author and guide computation through close, tacit understanding of movement.

[9] We should note that the robot was more able to follow the choreography because the notation was designed specifically for it. It would have been impossible for the robot to execute instructions designed for a human body, such as balancing or jumping.



```
  return stack(speechPat.speak(), pat.serial(38400));
}

var {action, to, part, by, dur} =
createParams('action', 'to', 'part', 'by');

window.a = slowcat(

 //Hello world. Please excute the following
instructions:

 //action('switch')
 // .part('head')
 //.dur(500)

  //action('reset')

 //action('move')
//  .part('waist')
//  .to (0.4)
```

```
1 strudel
)
var {action, to, part, by, dur} =
createParams('action', 'to', 'part', 'by');


window.b = stack(s("[bd ~] [~ ltt]mmt")
                  .chop(8,12)
                  .jux(rev)
                  .webdirt().slow(4)
                 )

stack(wind[illegible]window.b)
```

Figure 3: View from the audience of live coded instructions for movement, at the International Conference on Live Interfaces, June 2022. Photo: Ricardo Geraldes.

The future of this work includes workshopping a notational system that allows for further computational real-time notations focusing on movement for the robot as well as movement for screen dance in addition to our human dancers. We aim to continue jumping between fleshy and robotic movements to understand the relationship between them, and are already working to compare and contrast efforts between human and machine expression.

Developing notation through use

Responding to our first experimental performance, we are developing the next iteration of our notation. Our design process is now focussed on the movements of each of the individual servo-motors as curves over time, combined into whole gestures. Rather than a goal-directed approach (such as our initial experiment with inverse kinematics mentioned earlier), we begin with movement qualities of each individual motor, and then progress to combine those qualities to discover whole gestures that emerge when the three motors move together to form a gestalt, greater than the sum of its parts. These gestures then form the basis of our choreographic vocabulary.

The aim however isn't a fixed set of gestures that are then sequenced into a piece. Performing one gesture followed by another is one key way of combining gestures, but there are many more ways to combine them, and then to combine those combinations. For example, because a gesture is described as numbers changing over time, it makes sense to combine two gestures by simply adding them together, or multiplying one by another. Or, the gestures could be segmented, and then interlaced, one by another. These may seem like abstract operations, but are well grounded in the algorithmic underpinnings of the long history of craft, as illustrated in our earlier examples comparing weaving and bitfield patterns. Given suitable techniques for combining otherwise simple elements, creative ground is opened up to explore.

We also are interested in interactions between the human choreographer/performers and the robots within a contained performance. One way to facilitate this back and forth of movement is through the use of sensors to detect movement from the humans and map this to the robots. We are developing the use of accelerometers to shape both the movement of the robot during the performance, and the underlying tempo. This might affect the overall impulse of the robot's motion or just one of the three motors at a time, again allowing for combinatorial actions within the robot's choreography. This approach provides an opportunity to explore patterned movement not only on a single robot, but as live feedback between humans and robots.

Conclusion/Exhaustion

Ironically, automation is exhausting. As we already noted with reference to Bainbridge (1983), automated processes must be supervised by an expert, but the usual hands-off nature of automation means that their vigilance and expertise wanes. We have explored an alternative to automation, where live code, as higher-order notation, is placed in the same cybernetic loop as human and/or robotic movement. Here the code does not *control* the movement, but rather induces it by describing unpredictable (yet deterministic) patterned movements. The live coder responds to those movements by manipulating the code, to the point that the code follows from the movement, as much as the movement follows from the code.

What code *does* in this configuration is make an explosion of possibilities explicit. Language is combinational in that given a fairly small number of elements and an expressive language syntax, there may be a very large number of possibilities in which those elements can be arranged. When these language elements represent transformations of pattern, by combining those transformations together we find a practically inexhaustible supply of new patterns to explore.

When we share our live code manipulations then, we are sharing a kind of collective wealth that Mark Fisher argues that capitalism is set up to block: *"Real wealth is the collective capacity to produce, care and enjoy. […] This is Red Plenty [...] Everything for everyone. All of us first."* (Fisher 2018). We share code, and changes to it, in exploration of the pattern that results. Anyone can take a snapshot of that code, and make a minor adjustment to make something totally new. Scarcity is lost.

This generation of collective wealth from patterns is not new, but is the basis of much of traditional craft. For example a textile fragment such as the bronze/iron age card-woven fragment found in the salt mines of Hallstatt, may have a complex structure, but that structure may be 'read' to reveal the movement-code used to construct it (Griffiths et al 2022). So here we join a wider movement to reconnect contemporary technology with its basis in ancient craft, making that technology more open to change as cultural expression.

Discussion of technology, craft and exhaustion brings the industrial Luddite movement to mind, and live coders have already been compared to Luddites for valuing human craft over automation in their practice (McLean 2017). Indeed, live coding has been characterized by Colombian psychologist Camilo Andrés Hoyos Lozano (2022) as a technology of *post-work*, for its particular relationship with automation, its free/open source ethos, and its capability to restructure rules as they are followed. Live coder and researcher Alejandro Franco Briones (2023) sees similar possibilities for live coding in resisting marketisation and supporting emancipation.

But still, our project to date is limited by the conventional, off-the-shelf robotic technology that we have used, in that it is expensive, reliant on exploited labor, and reliant on supply channels made fragile by ongoing environmental collapse and resulting health emergencies. However, there is no shortage of electronics to be found in e-waste stockpiles, including motors. For our next steps then, we will look for ways to apply our work to repurposed servo motors, adjusting to an open hardware approach. Or if all else fails, we can still return to apply our notations solely to human movement.

References

Harlizius-Klück, Ellen. 2017. "Weaving as Binary Art and the Algebra of Patterns". TEXTILE 15 (2): 176–97. https://doi.org/10.1080/14759756.2017.1298239.

https://doi.org/10/fhdj8w

Blackwell, Alan F., Emma Cocker, Geoff Cox, Alex McLean, and Thor Magnusson. 2022. *Live Coding: A User's Manual*. MIT Press. https://doi.org/10.7551/mitpress/13770.001.0001.

Fisher, Mark. *k-punk: The Collected and Unpublished Writings of Mark Fisher (2004-2016)*, edited by Darren Ambrose. London: Repeater, 2018.

Franklin, Ursula. 1999. *The Real World of Technology*. 2nd edition. House of Anansi Press.

Griffiths, David, Alex McLean, and Amber Griffiths. 2022. "Digital is physical & a remote tablet weaving exploration." Then Try This. https://doi.org/10.5281/zenodo.7380868.

https://doi.org/10.1080/14759756.2017.1298239

https://andrewhugill.com/writings/Shifting%20meanings.html

Ingold, Tim. 2011. 'The Textility of Making'. In *Being Alive: Essays on Movement, Knowledge and Description*, 210–19. Routledge.

Podmore, Jono. 2020. *Jaki Liebezeit: The Life, Theory and Practice of a Master Drummer*. S.l.: Unbound.

Bainbridge, Lisanne. 1983. "Ironies of Automation." *Automatica* 19 (6): 775–79. https://doi.org/10/fhdj8w.

Cocker, Emma. 2014. "Live Notation - Reflections on a Kairotic Practice." Edited by Jerome Fletcher and Ric Allsopp. *Performance Research Journal* 18 (5).

———. 2017. "Weaving Codes/Coding Weaves: Penelopean Mêtis and the Weaver-Coder's Kairos." *TEXTILE* 15 (2): 124–41. https://doi.org/10.1080/14759756.2017.1298233.

deLahunta, Scott. 2002. Duplex/ ChoreoGraph: in conversation with Barriedale Operahouse. http://www.sdela.dds.nl/sfd/frankfin.html.

Eacho, Douglas. 2021. "Scripting Control: Computer Choreography and Neoliberal Performance." *Theatre Journal* 73 (2): 339–57.

Franco Briones, Alejandro. 2023. "Towards Another Transdiscipline: Art, Science and Emancipation as a Promise and Provocation for Live Coding." In . Utrecht, Netherlands: Zenodo. https://doi.org/10.5281/zenodo.7842097.

Harlizius-Klück, Ellen. 2017. "Weaving as Binary Art and the Algebra of Patterns." *TEXTILE* 15 (2):

176–97. https://doi.org/10.1080/14759756.2017.1298239.

Hugill, Andrew. 2020. “Shifting Meanings: The Fate of Words in Transdisciplinary Academia.” January 2020. https://andrewhugill.com/writings/Shifting%20meanings.html.

Laban, Rudolf von, and F. C. Lawrence. 1947. *Effort*. Macdonald & Evans.

Lozano, Camilo Andrés Hoyos. 2022. “Critical Subjectivity in Algorave’s Post-Work Practices.” *New Sociology: Journal of Critical Praxis* 3 (June). https://doi.org/10.25071/2563-3694.40.

McLean, Alex. 2017. “Lessons from the Luddites.” *Furtherfield* (blog). June 2, 2017. https://www.furtherfield.org/lessons-from-the-luddites/.

McLean, Alex, and Hester Reeve. 2012. “Live Notation: Acoustic Resonance?” In *Proceedings of International Computer Music Conference*, 70–75.

Roos, Felix, and Alex McLean. 2023. “Strudel: Live Coding Patterns on the Web.” In *Proceedings of the 7th International Conference on Live Coding*. Utrecht, Netherlands: Zenodo. https://doi.org/10.5281/zenodo.7842142.

Schiphorst, Thecla. 1993. “A Case Study of Merce Cunningham’s Use of the LifeForms Computer Choreographic System in the Making of Trackers.” Simon Fraser University.

Seth, RajyaLakshmi. 2017. “Oral and Written Traditions in Documentation of Dance Notation in Indian Classical Dances.” Nehru Memorial Museum & Library. http://www.indianculture.gov.in/research-papers/oral-and-written-traditions-documentation-dance-notation-indian-classical-dances.

Sicchio, Kate, Patrick Martin, Charles Dietzel, and Alicia Olivio. 2022. “Towards A Framework For Dancing Beyond Demonstration.” In *MOCO ’22: Proceedings of the 8th International Conference on Movement and Computing*. Chicago: ACM. https://dl.acm.org/doi/10.1145/3537972.3537981.

Sicchio, Kate, and Alex McLean. 2017. “Sound Choreography <> Body Code: Software Deployment and Notational Engagement without Trace.” *Contemporary Theatre Review* 27 (3): 405–10. https://doi.org/10.1080/10486801.2017.1343244.

Sulzman, Mona. 1978. “Choice/Form in Trisha Brown’s ‘Locus’: A View from inside the Cube.” *Dance Chronicle* 2 (2): 117–30. https://doi.org/10.1080/01472527808568723.