Creating a Digital Mirror of Creative Practice

Colin G. Johnson[0000−0002−9236−6581]

School of Computer Science
University of Nottingham
Nottingham, UK
Colin.Johnson@nottingham.ac.uk

Abstract. This paper describes an ongoing project to create a "digital mirror" to my practice as a composer of contemporary classical music; that is, a system that takes descriptions (in code) of aspects of that practice, and reflects them back as computer-generated realisations. The paper describes the design process of this system, explains how it is implemented, and gives some examples of the material that it generates. The paper further discusses some broader issues about the technological approach to building creative systems, in particular the advantages and disadvantages of building bespoke algorithms for generating creative content vs. the use of optimisation or learning from examples.

Keywords: music composition; computational creativity; optimisation; creative practice; computer composition; reflective practice; autoethnography; research by design

1 Introduction

The aim of this paper is to discuss work-in-progress in creating a computer system that imitates aspects of my artistic practice as a composer of contemporary notated music for (primarily) acoustic instruments. Call this system a *digital mirror*, as it attempts to "mirror" an understanding of artistic practice back to me. A related concept is the *digital twin* [10] in engineering—a computer system that replicates a physical system in detail. Working on this system has encouraged me to reflect on different means of creating artistic generative systems, and the various advantages of these and how they can be combined together.

What are the purposes of this digital mirror? The first is to act as a concrete way of reflecting on practice and techniques. Articulating and discussing aspects of practice in words is a powerful way of understanding and developing that practice. Trying to build an implementation of that technique requires yet more rigour, because code is an unforgiving medium, requiring each assumption to be spelled out clearly and unambiguously. When writing or speaking about an aspect of a practice, it is possible to rely on the reader/listener to fill in gaps and assumptions from their knowledge—or to engage in dialogue to resolve those ambiguities and gaps. By contrast, building a description of aspects of a practice in terms of code cannot rely on any interpretive skill or contextual knowledge from the computer.

The second is to act as a source of inspiration. By having a machine mirror back an interpretation of a creative practice, the creator can understand how clear that articulation is. These realisations, particularly when they produce unexpected results, can help to refine the creator's understanding of their practice. If a machine attempts to realise an articulated aspect of a practice, and produces unexpected content, then that can expose aspects that were not clearly articulated. Furthermore, these outputs—particularly ones that are somewhat unexpected—can act as sources of inspiration for the creator, and push their practice in new directions that might not have arisen from traditional reflection.

This paper begins (Section 2) with a discussion of the process of articulating a practice, together with a specific set of examples. It then moves on in Section 3 to discuss methods for creating a digital mirror, including a discussion of the advantages and disadvantages of four approaches: writing algorithms to create content, deriving content from statistical distributions, using optimisation to generate content, and generating content by imitating examples. Section 4 then describes the design and implementation of such a system and gives a number of examples of its outputs. Finally, section 5 concludes the paper with a summary, and directions for ongoing work.

2 Articulating aspects of a Practice

A common way in which to develop a creative practice is to describe the ideas that underlie that practice. This can involve articulating the the techniques and processes used, sources of inspiration and material, and the meaning or emotion that the practice is designed to invoke in an audience. In this paper, I am focusing on the first of these, the articulation of process and technique.

A positive way to think about such processes and techniques is that they are the way in which a particular creative individual realises their distinctive voice as practitioner. It is common to say that a particular practitioner (or a "school") has a particular voice or style. One way in which this cashes out is in the use of specific techniques to shape, process and arrange material. This can be at a number of scales. For musical composition, such techniques can range from methods of the large scale organisation of musical material in specific forms, down to the detailed organisation of musical notes. This is not the only way in which ideas of voice and style cash out. For example, the *subject* of an artform, what it is *about*, can be an important aspect. In more abstract artforms, such as the "music alone" [13] of purely instrumental music, the importance of techniques and processes become more prominent.

There are also negative aspects to such patterns. Overuse of a technique, or use of a technique with little reflection on the wider aesthetics of the piece, can become clichéd. Of course, reflection on and articulation of practice can be used to avoid, develop, or move on from techniques that have become stale.

This articulation is also part of a wider turn in creative practice towards seeing the creative process as a *research* process. that is, not just that creative practice "might on occasion *depend* on research" [8], but that it is a research

process in its own right [21]. Whether creative practice can be characterised in this way is controversial. Croft [8] argues that creative practice in musical composition lacks answerable research questions, clear research hypotheses, and a sense of generalisation from specific examples, and is therefore different to a research practice. Reeves [22] argues that insisting on a formulation of research that is focused on such a hypothetico-deductive method is too narrow a view of research, and that creative practice as research affords an opportunity to expand our view of research beyond a science-based paradigm. Similarly, Frayling [9] has emphasised that research in art and design areas can include research "through" creative practice, as well as "into" and "for" practice. An important part of any research process is making the objects of that research clearer through careful definitions and descriptions. This paper focuses on articulating these in the form of computer code.

2.1 Articulating my Practice

In my own practice as a composer of contemporary classical music for (primarily) acoustic instruments, I have developed a number of techniques. Often, such techniques have arisen out of abstractions from improvisatory practice or *bricolage*, particularly in the case of the practices that are concerned with detailed sequences of notes. These are then abstracted into techniques that are used and developed consciously. Larger scale structural elements arise less from such improvisatory processes, and more from reflection on the structure of composition, including what has been successful in my past work, and exemplars from elsewhere.

These techniques exist at various levels of granularity, from the details of note choice and the creation of textures through to large scale structural features. Here are a few prominent examples, with notated excerpts given in Figures 1 and 2.

Slapdash serialism of intervals. Musical lines created by successive notes, where over the course of the line, a large number of different intervals are found between successive notes (example in Figure 1a).

Layered and mutated melodies. Melodic lines, i.e. sequences of notes with small intervals (usually tones/semitones) between successive notes, often with a regular rhythmic pattern. These are layered on a number of instruments/voices, with the melodic pattern repeating but with mutations—changes of pitch, notes missed out, short gestures repeated. The layerings are uneven, with entries coming in at a variety of times relative to the other instruments (example in Figure 1b). Often, the melodies are locally tonal, but overall don't have a strong key centre.

Repeated chordal material. Repeated chordal material, sometimes antiphonal (bouncing back-and-forth) between two voices, to provide a static background against other material that is happening at the same time, or to provide a period of stasis in between two more active sections of music (example in Figure 1c).

Contrasted held material against shorter/moving material. Some instruments/ voices will hold a chord, single note, or a melodic or textural material. Typically, this will be static material, but sometimes a complex texture is used instead—if the texture is sufficiently complex, it becomes a single, unified sound. At the same time, there are short gestures—individual notes, chords, flurries of notes, or spoken text. An example is given in Figure 1d. Often there is a large difference in dynamics between very soft static material and very loud gestural material.

Trills and tremolo. Passage of trills (rapidly repeating pairs of notes), tremolo (rapid repetition of a single note), approximate trills, rapid repeated short scale passages, sometimes not accurately repeated (example in Figure 2a, which also illustrates passing use of slapdash serialism of intervals and repeated chordal material).

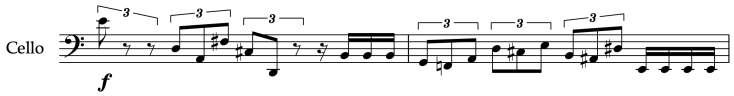
Large-scale approximate repetition. Large blocks of material return later in a composition; sometimes these are exact repetitions, sometimes they are distorted by changes of instrumentation, dynamics, or harmony; sometimes these repetitions are short extracts from earlier material, typically from the middle of a block of material from earlier in the piece.

Silent blocks and cutouts. One large-scale structural element is the use of long silences in the middle of otherwise coherent musical material, rather than to mark divisions between different sections of contrasting material (example in Figure 2b). This takes inspiration from ideas from Elliott Carter [3], particularly as realised in his First String Quartet. Relatedly, the repetition of material, but with blocks of notes cut out, perhaps just in some instrumental parts.

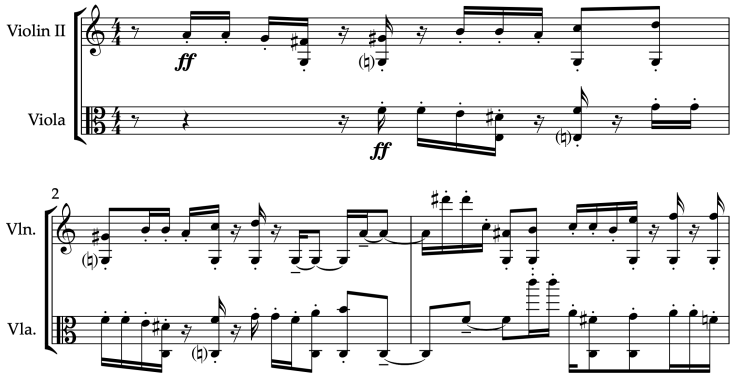
It should be noted that these articulations of practice are not necessarily accurate ones. In the midst of a complex process such as composition, it is easy to diverge from a specific practice that you have committed to; it is also notable that these are attempts to abstract from a complicated practice where material is not only influenced (consciously or not) by these practices, but also by reflective, listening processes and the qualia of listening. Furthermore, a single sonic event—say, a note—can be playing multiple roles at the same time: as part of a contrapuntal passage where different lines of music interact, as part of a harmonic structure, and as part of a larger scale structural block. Each of these different roles will impose constraints on groups of notes, meaning that they cannot fulfil their roles in these various techniques in a perfect way (not that that is always desired anyway; the techniques are a means to a musical end, not the end in itself). Indeed, this is one of the reasons for building a digital mirror—to see how the computer resolves these different constraints, and perhaps suggests new ways to combine them.

3 Methods of Creating a Digital Mirror

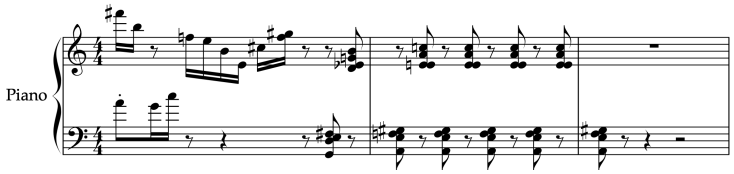
In creating a digital mirror of a creative practice, what techniques could be used? One approach is to train an AI system in an end-to-end way from a corpus of



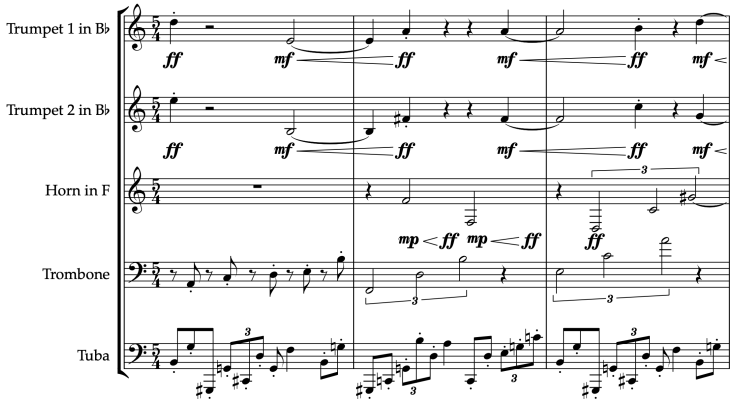
(a) Bars 10–11 of *"Her Fingernails Struck Carillons"* (2001) for small ensemble, cello part.



(b) Bars 1–3 of *Juxtapositions* (2020) for string quartet, violin II and viola parts.

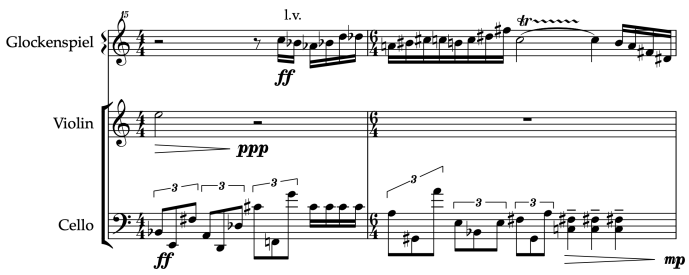


(c) Bars 29–32 of *In Medias Res* (2014) for solo pianist and ensemble, piano part.

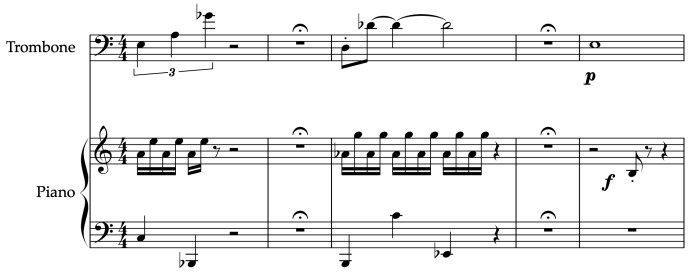


(d) Bars 51–53 of *Ballyhoo* (2017) for brass ensemble and organ, brass parts.

Fig. 1: Examples of Compositional Practice (1)



(a) Bars 76–77 of *"Her Fingernails Struck Carillons"* (2001) for ensemble, percussion, violin and cello parts.



(b) Bars 5–9 of *Five Glimpses into Alternative Universes* (2020) for trombone and piano.

Fig. 2: Examples of Compositional Practice (2)

examples—i.e. a corpus of whole pieces, not examples of the specific practices. In this approach, the various aspects mentioned above would not be represented explicitly in the code of the mirror, but would (perhaps!) be learned by the system from the corpus of examples. This is the approach taken by researchers such as Cope [7] in building systems that pastiche historical music.

This is not the approach taken here. Instead, the approach will be to create a number of computational systems that generate material that is aligned with the description of the kind of material that is generated in the practice. This distillation can be seen as akin to an expert system, but an important difference is that an expert system usually distills the knowledge of a number of experts, whereas in this system there is a single "expert" on the compositional practice. It is a mirror not to the practice as seen externally in the material provided to performance and/or to audience, but a mirror to the practice as articulated from within. A related piece of work is the painting robot AARON [5, 16], where the artist Harold Cohen worked over many years on a piece of painting software, modifying the software by reflecting on the successes, failures and serendipities of each version of the software.

There is a resonance here with the use of blackboard systems for poetry generation by Misztal-Radecka and Indurkhya [18]. In that system, various agents that encode expertise in different aspects of poetry (rhyme, metre, metaphor, etc.) take fragments of poetry from a common blackboard, modify that fragment,

then return it to the blackboard. In that work, the agents have a rather generic idea of the kind of work that they are trying to create; by contrast, in this paper the focus is on experts that articulate specific aspects of my own practice.

3.1 Algorithmic Methods for Generating Material

An important consideration in building systems of this kind is to consider the advantages and disadvantages of different kinds of processes. For the purposes of this paper, we will group generative processes into four categories:

Algorithmic Generation. In this kind of generative process, code is written that directly generates the material. This could be a deterministic process where the outcomes of the process are easy to predict, a process involving random choices at the micro-level, or an emergent process (whether deterministic or stochastic) where the results are harder to predict (i.e. generative art in the sense of Galanter [11]).

Distributional Generation. In this kind of generative process, material is generated from a statistical distribution. The origins of this are in the observation by Xenakis [29] that for a formal generative system of sufficient complexity, a similar psychoacoustic effect can be generated by sampling from a statistical distribution. As a naive example, consider serialist pitch organisation. As Xenakis [28] has noted, complex compositional frameworks are self-destroying: "what one hears in reality is nothing but a mass of notes in various registers" [28] (translation in [29]). This can be distinguished from the use of randomness in the Algorithm Generation process in that here the probability distributions are used at a macro level to describe large sections of material, whereas in the former randomness is used at a micro-scale to generate individual notes or gestures.

Optimisation. In this approach, the desired outcome is specified in a declarative way—the creator specifies what outcome is wanted, but not how to generate it. This could, for example, be by specifying a fitness function. The actual material is then generated by optimisation guided by that fitness function. These approaches can use statistical distributions, but via a generate-and-optimise process. For example, rather than sampling from a distribution,a measure is constructed of how accurately a particular example matches a distribution, and that measure is used as a fitness function in an optimisation algorithm such as a GA.

Corpus Imitation. In this approach, a number of examples of the use of the technique are collected, and this corpus is then used to train a machine learning algorithm to imitate and generalise from it.In Ritchie's [23] terminology, this is an "inspiring set". However, this inspiring set would not be of whole artworks, but instead of particular exemplars of the use of a specific technique. The idea would be to use machine learning to build models that act as experts on that specific aspect of practice. This is distinguished from the Distributional Generation in that in this approach, the distribution (or other model) is learned from a corpus of examples; whereas in the former, the distribution is crafted directly.

3.2 Example: Slapdash Serialism of Intervals

Consider the example of *slapdash serialism of intervals*, which was one of the techniques that I articulated above. In short, this means generating musical material where, over a decent period of time, the intervallic leaps between successive notes in an instrumental part are, roughly, equally distributed; there are the same number of minor thirds as there are perfect fifths as there are major sixths, etc. How would this material be generated in each of the four algorithmic methods described above? The focus here is on generating a single line of music.

Algorithmic Generation. Notes would be generated one-by-one in sequence, initially at random within a wide range. After each note is generated, the interval between it and the previous note would be stored, and a list of the intervals used built up as the musical line is developed. As the musical line develops, the probability of generating each note would vary depending on its interval from the previous note: if an interval had been heavily used up to that point in the line, the probability of choosing that note would be small; if the interval had been hardly used, the probability would be very high. As a result of this process, over a long musical line, the expected distribution of the intervals would be equal.

Distributional Generation. A uniform distribution of intervals would be created, the length of the musical line to be created decided, and a sample of that length generated at random from the distribution. This would then be used to generate a sequence of notes by starting from a random note, and moving through that list of intervals and using it to generate each note in turn. Note the similarities between this and the previous approach. The contrast between the two approaches is more pronounced when the stochastic, distributional approach is used to generate similar material to a deterministic algorithmic generative approach, as in the Xenakis examples, where sampling from a distribution of pitches, dynamics, note-lengths etc. is used to produce a similar musical effect as deterministic total serialism. There is another important difference. With the distributional approach, it is possible to combine two distributions in a fairly straightforward way (e.g. by forming a joint or conditional distribution). By contrast, in the algorithmic approach, it is not always clear how to combine two algorithms.

Optimisation. A population of random musiscal lines is generated by sampling uniformly from a chromatic scale of notes in a wide interval. This is the starting population in a genetic algorithm. The fitness function takes the sequence of notes, calculates the successive intervals between notes, counts each time a note has been used, and compares that to a uniform distribution of intervals using a root-mean-square error measure. This is then iterated until a musical line is discovered that has a low value for that error measure.

Corpus Imitation. A number of examples would be provided of musical lines that use this kind of interval distribution, which could then be used as the transition matrix in a generative Markov chain. Alternatively, a machine learning system such as an autoencoder [27] would be used, which takes the corpus of examples and learns to imitate their features.

3.3 Advantages and Disadvantages

All four of the above approaches can, broadly speaking, be used to generate similar material. This section discusses the advantages and disadvantages of each. Is not necessary to choose a single method for the development of a system. Particularly if the system architecture is designed as a collection of agents that generate, evaluate, and transform the material, each aspect could be generated using an appropriate approach.

In particular, this is one of the advantages of optimisation over algorithmic generation. In algorithmic generation, code is written to generate material that uses a particular technique or process. If we subsequently wanted to generate material that uses two processes at once, we would have to rewrite the code to generate material that aligned with both of those processes.

For example, consider trying to combine the *layered and mutated melodies* discussed above with the *trills and tremolos*. To create this in an algorithmic generative way, we would need to write a new piece of code that generated material that had that melodic aspect, whilst simultaneously including appropriate amounts of the trills, tremolos and repeated scale passages. This is not impractical, but it requires a lot of additional work and thought. By contrast, in the optimisation approach, what would be needed would be some way of bringing together a measure of "melodicity" and a measure of "trilliness" using multi-criterion optimisation. Then, an optimisation algorithm could search the space of possible note-sequences that included both of those aspects, without having to have written much new code other than a specification of how that multi-criterion optimisation would be done. Relatedly, a distributional generative approach can often combine different desirable properties by combining the underlying probability distributions.

A related advantage of the optimisation approach over the other approaches is that it is possible to use multi-criterion optimisation to create material that satisfies desirable criteria at different timescales. For example, one problem with the commonly used Markov models for algorithmic generation of musical or textual material is that, whilst the material can have desirable local structure, it lacks global coherence. Trying to achieve these two aims in a generative way is difficult, but again a multi-criterion optimisation approach has the potential to combine desirable properties on different time scales.

One disadvantage of an optimisation approach is that it is is difficult to discover material that requires a very precise placement of items. E.g., it would be time-consuming for an optimisation process to converge precisely onto a long trill between two notes, or to create a period of silence in the middle an otherwise busy texture. To create an approximate repetition of a whole passage of music later in a piece by optimisation of sequences of notes is even more challenging. These problems are multiplied if the optimisation algorithm is also optimising on other criteria. Generating such material is, by contrast, very easy using the algorithmic generation approach. A design pattern that has a lot of promise is to use algorithmic generation to generate this kind of material in the first place,

and then to use a fitness function to ensure that aspects of it are retained whilst other processes are acting on the material.

Another advantage of the optimisation approach is that different aspects of composition can be specified by different kinds of functions. As long as a function can return a fitness value, it can be incorporated into the system. By contrast, in the algorithmic generation approach, we have to choose up-front a "theory" of how our system is going to work. E.g., our theory might be that notes are generated successively, with each note value depending on a moving window of previous notes. This makes generation of certain kinds of material fairly straightforward (e.g. the *slapdash serialism* is easy to generate using this theory), but the large scale repetition of material isn't. The distributional generation is limited too—in that case we have already chosen what "theory" is going to be used, i.e. that of sampling from a distribution. If the desired process or technique can be realised as sampling from a probability distribution, then it is straightforward; otherwise, near impossible.

The choice of the word "theory" in the above discussion is designed to resonate with its use in the philosophy of concepts. Early theories of concepts rely on a single "theory" for all concepts [19]. For example, the prototype theory of concepts [24] is built on the theory that each concept is centred around a small number of prototypical examples of the concept, and that we assess whether an object is an example of that concept by closeness of that example to the prototype(s). By contrast, the more recent theory-theory of concepts [26] argues that each concept is associated with its own "theory". This fits amenably with the idea of fitness functions—each concept that we want to include in the generation has its own theory, realised in code. Multi-criterion optimisation does not require the same kind of theory to define each concept, only that it cashes out in the form of a scalar fitness measure. It would be interesting to try to drop even that requirement, and to see the process of multi-criterion optimisation to take the form of a "negotiation" between different desirable outcomes (e.g. based on computational argumentation [4]).

The final approach is learning from a corpus. One advantage that this has is that it can pick up on tacit features of the examples given. The other approaches are limited to those features that the system creator has articulated explicitly in the code, e.g. the generative code or the fitness function. By using machine learning from examples, aspects of those examples that are not obvious to the creator might be extracted; if the machine learning method used is an interpretable one, it may be clear what these features are.

4 Design, Implementation and Examples

Aspects of the above system have been implemented in a Python program, with the *Mido* package [17] being used for the generation of MIDI signals for playback via *Ableton Live* [1], and *MusicXML* [20] export being used to create notation which is realised in the *Sibelius* [25] notation system. A copy of the code can be downloaded from http://www.colinjohnson.me.uk/researchSoftware.php.

4.1 Structure of the System

The two main components of the system are *generators* and *drivers*. Generators are functions that create sequences of notes of a given length, and they (or expressions built from them) are used to create the initial population. A description of the generators available in the system are given in Table 1. Details of the implementations are given in the code mentioned above.

Table 1: Descriptions of Generators.

Generator	**Description**
Create Sequence	Creates a sequence of random notes in a given pitch-range (by default, midi notes 60-76), of random durations from 1–4 beats (quaver to minim).
Create Even-rhythm Sequence	Creates a sequence of random notes in a given pitch range (by default, midi notes 60-76), all quavers.
Create Trill	Chooses a random note from a given pitch range (by default, midi notes 60-76 and a second note a semitone or tone higher, and then creates a sequence of quavers alternating between those two notes.
Insert Gaps	Chooses a number (a user-specified parameter; 2 in the example below) of points in the sequence and adds a silence of between 20–25 quavers length.

Drivers are functions that take a sequence and return a number in the range $[0.0, 1.0]$, which measure some aspect of that sequence. These are designed so that fitness functions can be created either by using a single driver, or by combining drivers. A description of the drivers available in the system are given in Table 2.

Table 2: Descriptions of Drivers.

Driver	**Description**
Melodicity	A measure of how closely the distribution of intervals between successive notes in the sequence is concentrated around smaller intervals, in particular whole tones (a similar measure is explored in [2])
Slapdash Serialism	A measure of how closely the distribution of intervals between successive notes in the sequence is to a uniform distribution.
Trillicity	A measure of how closely the sequence matches with a trill passage.
Gappiness	A measure of how much the sequence contains large silent gaps.

The main evolutionary system works as follows. The user chooses a generator, and a fitness function which is either one of the drivers or an expression made up from the drivers. Each member of the population consists of a sequence of notes,

with the initial population being generated by the specified generator. A genetic algorithm is run, using tournament selection, elitism and a mutation operator only (no crossover). The mutation operator takes each note in the sequence and:

- If it is a note, with probability 0.1 changes the note value to a new value from a Gaussian distribution centred on the current note value and with s.d. 4 semitones, capped above and below by the MIDI notes 76 and 60.
- If it is a note and has not been changed by the above, with probability 0.1 changes it to a rest (if the "mutation to silence" parameter is true)
- If it is a rest, replaces it with a note with MIDI value drawn randomly in the range $[60, 76]$.

The parameters are summarised in Table 3. The parameter values vary between examples, as determined by empirical experimentation. These can be seen as "curated" examples, in that they are designed to show specific features that are of help in understanding the outputs of the system. Systems with a high "curation coefficient" [6] such as these, where the user has to explore a large number of runs or parameter settings, have been identified with systems that have low levels of autonomous computational creativity. For an application such as this, where the point of the system is to facilitate autoethnographic reflective practice, such concerns are less. The point of such a system is not to autonomously generate imitations and pastiches of the human creator's work as such; instead, this exploration of parameter space is part of that reflective process.

Table 3: Genetic Algorithm Parameters.

Example	**Sequence Length**	**Population Size**	**Max Generations**	**Tournament Size**	**Elitism**	**Mutation to Silence?**
Example 1	50	1000	100	7	no	no
Example 2	50	1000	100	7	no	no
Example 3	50	50	300	7	yes	yes
Example 4	50	50	1000	7	yes	yes

The system can be used to start from random sequences of notes, and to evolve towards a desired combination of features by specifying a fitness function that combines appropriate drivers. Alternatively, it can be used to create a transition between two kinds of material; as discussed by Magnus [15], one interesting way to use genetic algorithms (or other AI techniques) is to play out the evolutionary process as a performance; what Johnson [12] has called a "seeds and targets" approach.

4.2 Examples

This section of the paper gives a number of examples from the digital mirror. These are a mixture of examples to demonstrate particular points, and a final example that shows how multiple fitness functions can be brought together.

Example 1. This example tests the idea discussed earlier of whether it is difficult to use evolution to evolve an exact trill. The generator used in this experiment is *evenSequence*, which generates random notes in a certain interval, and the driver is *analyseTrillicity*, which measures how close the sequence is to a trill. For simplicity, we turn off the mutation that makes a note into a rest for this example. Samples from the 20th and 100th generations are shown in Figure 3. There is some aspect of trill, but it has not evolved a perfect trill.

Example 2. This uses the generator *generateTrill* so that the initial population consists of trills on a randomly chosen note. The driver is *analyseMelodicity*+ *analyseTrillicity*, that is, we are trying to keep the core trill behaviour, whilst adding melodic aspects to it. As we can see in Figure 4a, even after 10 generations we can see a good combination of short melodic phrases whilst retaining the trill behaviour; after 100 generations a different example, but with largely similar characteristics (trill passages then some melodic variation) has emerged.

Example 3. The generator used here is *evenSequence* combined with *insertGaps* which inserts silences. The fitness function moves this random note distribution towards a more "melodic" distribution, whilst also retaining those gaps. After 100 generations (Figure 5, the larger gaps have been retained, but the note patterns have a more melodic feel than the initial random note choices.

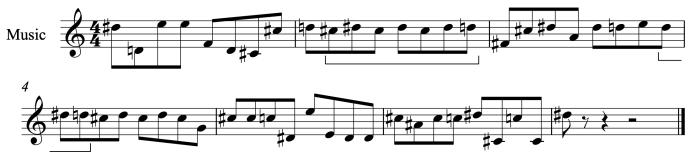
Example 4. This shows a more complex situation, where the generator used in is again *evenSequence* combined with *insertGaps*. However, the fitness function combines retaining gaps, creating trills, and creating a "melodic" distribution—all of which are retained/created in the example (Figure 6).

5 Conclusions and Ongoing Work

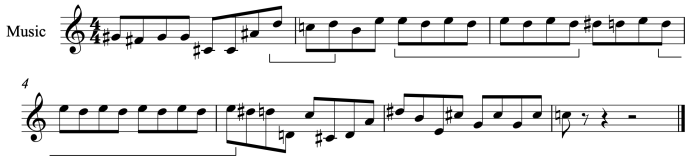
This paper has presented ongoing work on an attempt to make a "digital mirror" to reflect my articulations of compositional practice. Creating this has forced me to clearly articulate some of my processes, because code is an unforgiving medium where it is necessary to formally state assumptions. The aim of this is not focused on generating whole pieces of music, but to facilitate reflection.

This is an ongoing project, and there are a number of future developments. One would be to expand the optimisation techniques used, in particular using crossover and multi-criterion optimisation. In terms of musical development, there is much to be developed in terms of dynamics and rhythm, which are very limited in the current version. Another limitation is that the current version is that there is a single line; it would be interesting to explore both multi-line notated music, and multiple interacting musical agents (as explored by Lewis [14]).

One key place where the digital mirror falls down is that it—obviously—doesn't replicate the conscious qualia of listening. Musical techniques almost always admit some *choice*, whether of note-level details, or high-level parameter choice. In a digital mirror, these choices are typically made randomly, with

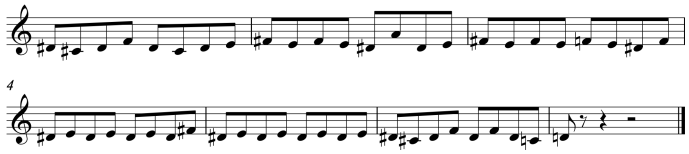


(a) Trying to evolve a trill: after 20 generations.

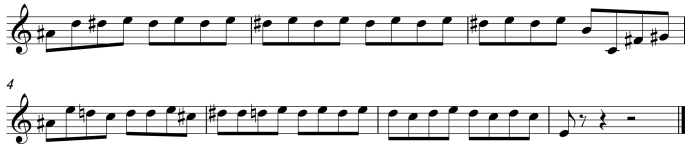


(b) Trying to evolve a trill: after 100 generations.

Fig. 3: Trying to evolve a trill (trill fragments bracketed).



(a) Making a trill more melodic: after 10 generations.



(b) Making a trill more melodic: after 100 generations.

Fig. 4: Making a trill more melodic.

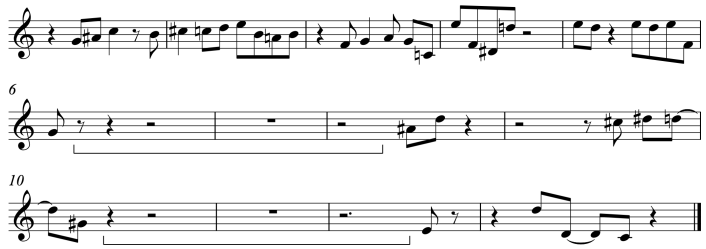


Fig. 5: Evolving from a random sequence of notes to a more "melodic" sequence, whilst retaining gaps: after 100 generations (gaps bracketed).

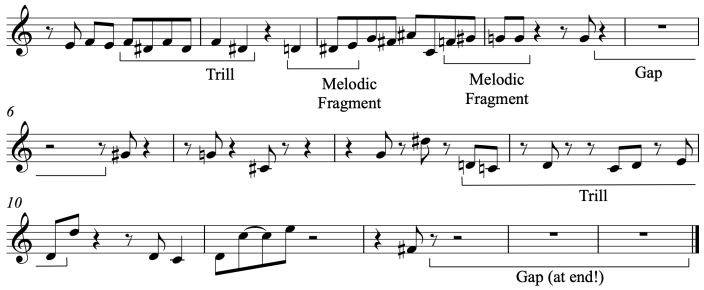


Fig. 6: Evolving with a more complex fitness function: after 800 generations (notable features bracketed).

some filtering out by the fitness function. But, this doesn't replicate the reflective process of *listening* to a line of music, and getting a musical understanding. As Croft [8] puts it, there are plenty of ways of "solving problems" about creative practice, but in the end what is being sought are "striking or idiosyncratic *musical* solutions to problems of *musical* material that arise only during the process of composition". A merely formal solution to the question is not sufficient. Perhaps some kind of interactive evolution could afford some progress.

As a personal reflective process, I have overall found this useful. Formalising processes I thought I was using caused me to revise my understanding; e.g. I realised that what I was remembering as primarily trills and tremolos were more commonly rather rougher repetitions of short scale-fragments. In terms of the material that the mirror "reflected back", one example that surprised me was how short the material could be in between silent blocks. A couple of times the machine generated a passage consisting of just a beat or two of sound between long silences, and I found this musically effective. This is not something that I would have thought of without this. Perhaps there is future work to be done in this broad idea of using code as part of reflective practice and autoethnography.

Aside from the design and development of system, the paper has also considered the kinds of methods used; this is likely to be of interest to a wider constituency of people working on creative systems. In particular, the contrast between four methods of generating material in such a system has been discussed.

References

1. Ableton Live, https://www.ableton.com/en/live/, visited Nov. 2020
2. Calderon Alvarado, F.H., Lee, W.H., Huang, Y.H., Chen, Y.S.: Melody similarity and tempo diversity as evolutionary factors for music variations by genetic algorithms. In: Cardoso, F.A., Machado, P., Veale, T., Cunha, J.M. (eds.) Proceedings of the 11th International Conference on Computational Creativity. pp. 251–254. Association for Computational Creativity (202o)
3. Carter, E.: The time dimension in music. In: Bernard, J. (ed.) Elliott Carter: Collected Essays and Lectures 1937–1995. pp. 224–228. University of Rochester Press (1997)

4. Cocarascu, O., Toni, F.: Argumentation for machine learning: A survey. In: Baroni, P., et al. (eds.) Computational Models of Argument: Proceedings of COMMA 2016. pp. 219–230. IOS Press (2016)
5. Cohen, H.: The further exploits of AARON, painter. Stanford Humanities Review **4**(2), 141–158 (1995)
6. Colton, S., Wiggins, G.: Computational creativity: The final frontier? In: de Raedt, L., Bessiere, C., Dubois, D., Doherty, P. (eds.) Proceedings of the 20th European Conference on Artificial Intelligence. pp. 21–26. Amsterdam (2012)
7. Cope, D.: Computers and Musical Style. Oxford University Press (1991)
8. Croft, J.: Composition is not research. Tempo **69**(272), 6—11 (2015). https://doi.org/10.1017/S0040298214000989
9. Frayling, C.: Research in art and design. Royal College of Art Research Papers **1**(1) (1993–94)
10. Fuller, A., Fan, Z., Day, C., Barlow, C.: Digital twin: Enabling technologies, challenges and open research. IEEE Access **8**, 108952–108971 (2020). https://doi.org/10.1109/ACCESS.2020.2998358
11. Galanter, P.: What is generative art? complexity theory as a context for art theory. In: Proceedings of the Sixth International Conference on Generative Art (2003), https://www.generativeart.com/on/cic/papersGA2003/a22.pdf
12. Johnson, C.G.: Fitness in evolutionary art and music: a taxonomy and future prospects. International Journal of Arts and Technology **9**(1), 4–25 (2016)
13. Kivy, P.: Music Alone: Philosophical Reflections on the Purely Musical Experience. Cornell University Press (1991)
14. Lewis, G.: Too many notes: Computers, complexity and culture in *Voyager*. Leonardo Music Journal **10**, 33–39 (2000)
15. Magnus, C.: Evolutionary musique concrète. In: Rothlauf, F., et al. (eds.) Applications of Evolutionary Computing. Lecture Notes in Computer Science, vol. 3907, pp. 688–695. Springer Berlin / Heidelberg (2006)
16. McCorduck, P.: AARON's Code : Meta-art, Artificial Intelligence, and the work of Harold Cohen. W.H. Freeman (1991)
17. Mido: MIDI objects for Python, https://mido.readthedocs.io, visited Nov. 2020
18. Misztal-Radecka, J., Indurkhya, B.: A blackboard system for generating poetry. Computer Science **17**(2), 265–294 (2016)
19. Murphy, G.L.: The Big Book of Concepts. MIT Press (2004)
20. MusicXML, https://www.musicxml.com, visited Nov. 2020
21. Pace, I.: Composition and performance can be, and often have been, research. Tempo **70**(275), 60—70 (2016)
22. Reeves, C.: Composition, research and pseudo-science: a response to John Croft. Tempo **70**(275), 50—59 (2016)
23. Ritchie, G.: Assessing creativity. In: Proceedings of the AISB Symposium on Artificial Intelligence and Creativity in Arts and Science. pp. 3—11. AISB Press (2001)
24. Rosch, E.H.: Natural categories. Cognitive Psychology **4**, 328–350 (1973)
25. Sibelius, https://www.avid.com/sibelius, visited Nov. 2020
26. Weiskopf, D.A.: The plurality of concepts. Synthese **169**, 145–173 (2009)
27. Welling, M., Kingma, D.P.: An introduction to variational autoencoders. Foundations and Trends in Machine Learning **12**(4), 307—392 (2019)
28. Xenakis, I.: La crise de la musique sérielle. Gravesaner Blätter **1**, 2–4 (1955)
29. Xenakis, I.: Formalized Music: Thought and Mathematics in Composition. Indiana University Press (1971)