

Protocols and the irreducible traces of embodiment: the Viterbi algorithm and the mosaic of machine time

Adrian Mackenzie

Institute for Cultural Research

Lancaster University, LA1 4YD, UK

a.mackenzie@lancaster.ac.uk

May 2005

Protocols and the irreducible traces of embodiment: the Viterbi algorithm and the mosaic of machine time

Introduction: machine time is non-lived?

Machine time, it has often been argued in critical thought, is inhuman or non-lived time. Machine time is a relatively abstract, taken for granted term that refers to the speed at which mechanisms and devices carry out operations. Usually, responsibility for the deleterious and positive effects of technological speed is attributed to the high rate at which movements are carried out. For much critical theory, this ever-increasing rate (expressed prosaically say in CPU speeds used to advertise computers or in Moore's Law) reduces temporal complexities, memory and subjective experience to attenuated abstractions, to non-lived spaces and times, to intervals and orderings that are inimical to human lifeworld structures. Adverse reactions to machine time are marked particularly strong in the domain of media and images (Virilio on autonomization of perception is a good example). This chapter argues that it would be useful and perhaps important to reconsider machine time in less abstract terms. This involves finding middle ground between the temporality of technologies as material orderings of movement and the temporal flows of subjective experience. Through a case study of a specific telecommunications algorithm, it suggests seeing machine time as a mosaic of relations and orderings of actions brought into proximity.

One symptom of the aggregate or mosaic nature of machine time can be found in the very proliferation of different timings attached to contemporary computing and information technologies. They include seek time, run time, read time, access time, available time, real time, polynomial time, time division, time slicing, time sharing, time complexity, write time, processor time, hold time, execution time, compilation time, and cycle time. While some of these timings are related (for example, read and write time), many are unrelated or antagonistic to each other (for example, real time

and polynomial time). The relations between different timings are heterogeneous. While compilation time and execution time of a fragment of software may be closely associated for a programmer, the time complexity of a whole system comprising many different processes may be most relevant for people using networked computers. Machine time has many interstices, and they have implications for how we think about changes in perception, experience, movement, embodiment sociality and meaning associated with contemporary telecommunications and information networks. Attending to the constitution of contemporary machine would both continue long-standing attempts (beginning with Husserl's phenomenology) to find ways of re-activating the forgotten sense of the life-worlds we inhabit and bring into dialogue some of recent work in sociology, social studies of science and geography, new media and cinema studies that has addressed these changes in terms of structures of feeling, networks of actors, spaces and times of calculation , formal properties of media (moving images in particular ,) and new forms of sociality].

Framings of machine time

The study of machine time is difficult. How do we frame the time of machine operations? This question lies underneath much sociological and philosophical thought from the late nineteenth century onwards, when electric media, the physical and life sciences together started to challenge the verities of perception and action in new ways. Two related responses appear in recent accounts. For media theorists such as Friedrich Kittler, machine time falls below the thresholds of consciousness and thereby withdraws into the background of average everyday life: '[w]hat remains a problem is only recognizing these layers [of software], which like modern media technologies in general, have been explicitly contrived to evade perception' . This withdrawal is very much a structural feature of systems of control, information and communication, and permits, among other things their annexation of everyday communication as forms of property . In a broadly similar vein, media theorists such as Manovich , Rodowick , Lunenfeld and Cubitt understand digital media in terms of the formal properties and attributes that rupture habits of communication, perception and representation. In relation to time, such changes can only radically undermine the foundations of subjective experience in time consciousness.

For recent theorists such as Mark B.N. Hansen, the task is not to affirm the way in which

information processing evades perception, but to understand the challenge it poses to our understanding of embodiment, communication, and time more generally:

Viewed in this way, the digital era and the phenomenon of digitization itself can be understood as demarcating a shift in the correlation of two terms: media and body. Simply put, as media lose their material specificity, the body takes on a more prominent function as a selective processor of information .

In Hansen's terms, machine time does not simply evade perception. It is a radical restructuring of perception as selection from fluxes of information (or images in his analysis). In particular, 'machine time can, in some sense, be said *to enlarge the frame of the now itself* . In Hansen's case, a relatively complex neuro-phenomenological account of the constitution of time-consciousness ensues.

Across this range of responses, a relatively abstract characterisation of machine time remains intact. Even in his deeply interesting combination of neuro-phenomenology with concepts of embodiment and affect, Hansen's account of machine time remains relatively a-social. All complexity, dynamism and potential is again located in 'the body', which takes on 'a more prominent function'. Machine time, as a contingent effect generated by large numbers of machines connected together in materially and socially heterogeneous networks and infrastructures, remains unanalysed. The problem, therefore, is to find ways of articulating the uneven, mixed timings that emerge within real networks of inter-operating systems, and to understand these timings as living-non-living syntheses, not as confrontations between the living subject and dead, deadly machines. Machine time in networked, distributed information technologies is a truly complex assemblage, with many different interfaces, points of intersection and slippage, competing dynamics and intensifications. In a somewhat different context, Elspeth Probyn asks:

[H]ow we can cross over from the solitary space-time of individual categories in order to renew a critical emphasis on the proximity of sites. . Living in the space-time of a world rendered local, means that we have the capacity to be intimately confronted with the implications of our actions.

The key issue in this passage concerns the shift from 'solitary space-time' to 'proximity to sites,'proximity to others, and this curious, slightly awkward idea of intimately confronting the implications of our own actions. This shift is doubly relevant for an analysis of machine time. For on the one hand, contemporary communication networks are largely put in the service of forms of proximity (in time and space). Here are

located all the effects of real time media and telethesia so widely discussed in critical theory in recent decades. On the other hand, proximity refers to the clustering together of different times and spaces in composites that 'implicate' actions with each other. On this latter point, sociological studies of the time and space of global financial trading systems indicate how temporally complex the implication of actions can become .

Algorithm as articulation

As a preliminary crossing out of the solitary category of machine-time, we could think of some of the different conventional, artificial, inorganic, living, social, and semiotic materials that come together today in computing devices such as mobile phones, and wireless networking equipment. In such equipment, machine time emerges from specific articulations of materials in relations on widely varying scales - semiconductor fabrication, network protocols, licensing agreements, quotidian rhythms of work, branding, marketing, travel and consumption (see for instance, Donald Mackenzie's account of the relation between the design of nuclear test simulation and the working day of high-energy physicists), the implementation of standards, protocols, the marketing of network bandwidth to individuals, institutions and corporations, the effect of diurnal rotation of the earth on the electromagnetic spectrum, etc. Everywhere in this melange, articulations of different material constitute contemporary machine time. It only takes one particular algorithm to show this composite texture.

Algorithms are often seen as the abstract heart of software and digital technologies. Given that software has gradually taken shape as a new kind of substance, halfway between matter and mind, and has propagated throughout the matrices of everyday life , analysis of algorithms has a certain importance. In some accounts, for instance Steven Wolfram's cosmology , algorithms serve as ontological foundations of the universe. In many other accounts of technology, information or network society, algorithms are seen as all the same. From the perspective of humanities and social sciences, algorithms, like mathematics and geometry, belong to a world-view that quantifies, orders and sequences events and movement. However, as this chapter will argue, to treat algorithms as a general expression of mental effort, or, perhaps even more abstractly, as process of abstraction, is to lose track of proximities and relationalities that algorithms articulate. Regardless of how central algorithms are to mythologies of computation, information, economy, life or universe, they are rarely discussed in

themselves and rarely attended to as objects of analysis.

When they are examined, for instance by patent attorneys, algorithms are treated as the mechanical expression of mental effort. For instance, the several hundred patents currently associated with an algorithm discussed at length below, the Viterbi decoder algorithm are all judged as embodying singular applications of human individual or collective intelligence. The creation of value, particular intellectual value, is the foundation of intellectual or 'vectoral' forms of property . Algorithms are a key object of ownership. Current disputes over software patents hinge on the mode of existence of algorithms. In these disputes, the issue concerns the creation of algorithms, and how that creation can be made over into a form of property. Of particular interest is how the creation of an algorithm occurs as a mental act. What distinguishes algorithms from other forms of invention? The connection to be made here concerns algorithms as embodiments of mental effort or thinking. Considering how algorithms come into being and evolve, it is hard to deny some contribution from mental effort. The thought embodied in an algorithm does not simply represent a pre-existing object, nor create a new thing. The philosopher John Dewey suggested that thinking occupies a temporally intermediate reconstructive position which brings out some order in a discordant situation . It is hard to argue with the idea that algorithms embody highly refined, often formalised repetitions, but these repetitions themselves accomplish a movement that reduces discord. Algorithms handle repetition in interestingly variable ways. Furthermore, the repetition of algorithms themselves, the ways in which they are imitated and altered as they circulate is in no way mechanical or abstract. Algorithmic functions shift and refract as they repeat movements. They move towards the fringes of bare mechanical repetition, where hidden states, indistinct feelings and sensations cluster around the 'bright nucleus' of intellect and action that mechanism represented for Bergson .

Viterbi algorithm – evolution, telecommunications and language processing

The Viterbi algorithm runs across telecommunications, new media, and a contemporary incarnation of biopolitical knowledge project, bioinformatics. The algorithm dating from 1967 is widely used in telecommunications networks to create and discover position and sequence in flows of signals. It has become a fundamental element in wireless, satellite

and space communications. Andrew Viterbi, a now retired telecommunications engineer, designed the algorithm and started a company that designs and fabricates semiconductors based around the algorithm. In telecommunications applications, these chips enable satellite, cellular phone and wireless networks to communicate despite high levels of electromagnetic noise. In bioinformatics, the Viterbi algorithm is currently used to find weak similarities and evolutionary kinship between protein or amino acid sequences within families of functionally related biomolecules .

The Viterbi algorithm deals with situations where the generation of sequential system states cannot be directly observed. This could be because they are lost in evolutionary history (bioinformatics), or because a signal transmitted in the crowded electromagnetic environment of a city (or interplanetary space) is noisy or variable. According to the account given in Wikipedia, 'the Viterbi algorithm was originally conceived as an [error-correction](http://en.wikipedia.org/wiki/Error-correction) scheme for noisy digital communication links, finding universal application in decoding the [convolutional codes](http://en.wikipedia.org/wiki/Convolutional_code) used in [CDMA](http://en.wikipedia.org/wiki/CDMA) and [GSM](http://en.wikipedia.org/wiki/GSM) digital cellular, dial modems, satellite and deep-space communications, and [802.11](http://en.wikipedia.org/wiki/802.11) wireless LANs. It is now also commonly used in [information theory](http://en.wikipedia.org/wiki/Information_theory), [speech recognition](http://en.wikipedia.org/wiki/Speech_recognition), [keyword spotting](http://en.wikipedia.org/wiki/Keyword_spotting), [computational linguistics](http://en.wikipedia.org/wiki/Computational_linguistics), and [bioinformatics](http://en.wikipedia.org/wiki/Bioinformatics). For example, in speech-to-text speech recognition, the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal' . In general, the algorithm finds the most likely series of hidden events that could have given rise to the observed events. The fact that the algorithm merits an article in Wikipedia attests something of its significance as an embodiment of thought and an organisation of movement.

The basis of the Viterbi's application to speech-processing, bioinformatics, and

telecommunications, or to language, life and media, to use more abstract terms, hinges on the idea of finding the most probable *hidden states* that would account for the currently observed behaviour in a system. The movement accomplished by the algorithm is reconstructive in Dewey's sense. It realigns a problematic situation. For instance, in a 802.11b wireless network, a GSM cellular telephone network, satellite or the Cassini Saturn probe, data to be transmitted is coded using a particular approach called 'convolutional coding'. In convolutional coding, the computational processing capacity of the transmitter and receiver is used to compensate for noise and errors produced in the transmission channel. Intensified movements in transmitter and receiver compensate for disturbances or errors arising from what cannot be made fully part of the system, the medium of propagation of signals. Convolutional codes get their name from the way in which they base what they transmit at the current point in time on what has been transmitted earlier. They begin to build summaries of what has preceded the current moment into each moment. The 'convolution' consists in this turning inwards of encoding to include information about what was transmitted before. Each packet, datagram or frame represents not just information, but a relation to other information. A convolutional code 'enlarges the frame of the now itself' in specific way, and in relation to particular forces.

Interpreting algorithms: imperative, brute force and abstraction

However, does not this enlarged now still remain mechanical? How can an algorithm be understood as anything other than mechanical? The prevalent interpretation of algorithms sees them as program, as sets of instructions to be carried out in sequence in order to accomplish some task. Presenting an algorithm as a recipe means treating it as *imperative*. This means that the algorithms lend themselves to being seen as commands that are carried out mechanically. Other interpretations or formalizations of algorithms that appear much less mechanical are well-known although less commonly used. It would be interesting to analyse the social dynamics of formulations of algorithms. For instance, 'functional programming' is based on the recursive evaluation of expressions, something that appears far less mechanical, yet less widely-used. (However, on this score, see for an entrepreneurial description of how an early e-commerce business used functional programming to out-accelerate competitors in the dotcom boom.)

Computer science textbooks usually classify algorithms in a relatively small number of

categories based on how the algorithm works. Divide and conquer, search and enumeration, probabilistic, heuristic, and dynamic programming are four main categories. Most books on algorithms are actually catalogues of algorithms combined with explanation and examples. The Viterbi algorithm, for instance, belongs to the class of dynamic programming algorithms. There are two frames that unify algorithms as a field of operational practice. The first is a limit notion of 'brute force', where the 'programmer relies on the computer's processing power instead of using his or her own intelligence to simplify the problem'. Brute force, at least from the perspective of programmers, lacks grace or abstraction. However, brute force may be the best technique when a smart solution would cost a lot of programming time. In his recent work on 'qualcalculation', argues that the availability of large amounts of computer processing capacity distributed in the built-background of everyday life is supporting new apprehensions of space and time based on movement. From an algorithmic perspective, the sheer availability of processing power means little. The question is how to order and select possible operations so as to create a consistently concatenated set of movements. In this selection, the key question is a relevant abstraction that responds to the needs of a situation.

A second overarching abstract perspective on algorithms as forms of movement is offered by computational or algorithmic complexity theory. This body of mathematical theory specifies the computational time and space in which the algorithm runs, and summarises using a specific notation, the big-0 notation. Knowing the complexity of an algorithm can help decide whether it is feasible to run it given the speed and memory of the system. Choosing or creating a different algorithm can drastically alter the temporality of computation. For instance, the flow of sensation and movement associated with contemporary action realtime games relies intimately on algorithmic orderings found in graphics calculations. Brute force approaches to animation do not work. The proofs and formalisms of algorithmic complexity theory express the time and space of any particular class of algorithms, and part of the working knowledge of programmers is some awareness of how different algorithms affect the time and space of computation.

Fold in time: stochastic processes and convolution

In terms of algorithmic complexity theory and computer science as operational

knowledges of machine time, we can treat an algorithm as a framing of time and space, as the creation of a duration. For instance, in convolutional coding, a practical folding of time occurs. It takes us away from algorithm understood as a linear sequence of steps to be carried out mechanically, to algorithms conceived as establishing relations between things that are disjointed, by concatenating events in paths. Convolutional codes are executed by software, and more often by VLSI (Very Large Scale Integrated) chips fabricated for use in cellphones, wireless networks and other communication equipment. But the implementation of the algorithm in silicon does not necessarily imply that the algorithm is purely mechanical or that it is governed by repetition. Rather implementations changes the texture of mechanism in certain ways. The power of the imperative style resides in its promise of exact repetition. This is what makes it mechanical - the specification of what is to be repeated. However, as Henri Bergson argued,

Repetition is only possible in the abstract: what is repeated is some aspect that our senses, and especially our intellect, have singled out from reality, just because our action, upon which all the effort of our intellect is directed, can move only among repetitions

Convolutional codes pose an obstacle to any reading of algorithms as repetition because they assume that events are not fully determined and therefore not definitively repeated. This begins to undermine a mechanistic understanding of algorithms as repetition. Different vectors of repetition interpenetrate. The Viterbi algorithm works with convolutional codes that themselves grapple with *stochastic* processes. These are processes in which the next state of the system is not fully determined by the previous state. A snakes and ladders game is a stochastic process because the next move is partially determined by a dice roll. It is the stochastic character of the Viterbi algorithm that both affects the qualitative effect of repetition, and alters the terrain on which machine time moves.

Probabilistic processes have been of great interest ever since population became a thinkable entity in the nineteenth century, something which could be counted and controlled . Probability and randomness have a number of different conceptual layers in the Viterbi algorithm. Firstly, the algorithm regards all signals, sequences or systems in probabilistic terms. In particular, it treats them as a Hidden Markov model. A Markov model is a way of understanding stochastic processes. It models situations by assuming

that the next state or event in the system is dependent only on the present state, not past states, and assuming that the most probable next state can be known. For instance, a game of snakes and ladders can be modelled as a Markov process because the next move depends entirely on the present state of the board, and a roll of the dice. (More formally, the conditional probability distribution of next moves in snakes and ladders depends on a current state and roll of the dice.)

Every algorithm, apart from so-called 'brute force' algorithms, contains a twist or kink that affects the flow of the computational process. The Hidden Markov model is one twist or kink at the heart of the Viterbi algorithm. It treats the received signal as a set of states that correspond to a Markov model that cannot be observed directly. Via the Hidden Markov Model, the Viterbi algorithm turns the communication situation inside out. The combination of a known and finite set of system states and probability is turned inside out by the algorithm because it treats the Markov model as hidden. The object of making a Hidden Markov Model is to deduce the most probable sequence of internal states that could have given rise to the observed sequence of signals.

We can begin to see how this twist might be useful in communication systems.

Something blocks access to what is actually happening in the situation. In telecommunications, including satellite, cellular and wireless network communications, many different kinds of interference and noise affect the transmission of digital signals. The task is to work out the most probable sequence of signals that could have given rise to the observed signals. Again, counter to the images of mechanical determinism sometimes associated with digital technologies or information systems, the algorithm engages with and provides an elaborate figure of an unpredictable and intrinsically dynamic environment. It moves to the fringes of mechanical action, where similarity or repetition begins to blur.

The sociality of an algorithm

Where does this convoluted displacement of repetition come from? What has happened to machine time to contort it into such unfamiliar, uncertain patterns of operation and movement? A simple explanation of the core of algorithm, again drawn from the Wikipedia Encyclopaedia provides one clue:

Assume you have a friend who lives far away and who you call daily to talk about what each of you did that day. Your friend has only three things he's interested in:

walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day. You have no definite information about the weather where your friend lives, but you know general trends. Based on what he tells you he did each day, you try to guess what the weather must have been like.

You believe that the weather operates as a discrete [Markov chain](http://en.wikipedia.org/wiki/Markov_chain) [<http://en.wikipedia.org/wiki/Markov_chain>](http://en.wikipedia.org/wiki/Markov_chain). There are two states, "Rainy" and "Sunny", but you cannot observe them directly, that is, they are *hidden* from you. On each day, there is a certain chance that your friend will perform one of the following activities, depending on the weather: "walk", "shop", or "clean". Since your friend tells you about his activities, those are the *observations*. The entire system is that of a hidden Markov model (HMM).

You know the general weather trends in the area and you know what your friend likes to do on average.

...

You talk to your friend three days in a row and discover that on the first day he went for a walk, on the second day he went shopping, and on the third day he cleaned his apartment. You have two questions: What is the overall probability of this sequence of observations? [{Wikipedia, 2005 #693}](#)

[<http://en.wikipedia.org/wiki/Viterbi_algorithm>](http://en.wikipedia.org/wiki/Viterbi_algorithm)

The explanation is simple and, symptomatically, refers to an everyday situation. It renders the core of the algorithm as a social situation. It renders the Viterbi algorithm thinkable as a set of questions about the weather. The problem of re-constructing how the weather turned out somewhere else hardly seems important, but it exemplifies the movement that Dewey ascribes to thinking. However, it embraces a complication. A social interaction (calling a friend to find out what they have been doing) furnishes the basis for implementing a form of controlled movement (deducing the weather) that is not itself intrinsic to that interaction. This aspect of the Viterbi algorithm, I suggest, *introjects* a communicative relation from everyday life . An introjection of sociality into the organisation of repetition begins to break down any simple opposition between machine and human time.

Dynamic programming and the logistics of intensive movement

In classifications of algorithms, the Viterbi algorithm is broadly regarded as a 'dynamic programming' algorithm. This second major formal aspect of the Viterbi algorithm stems from operations research, a field originating in WWII logistics. Computer science textbooks usually contain at least one chapter on dynamic programming. Dynamic programming addresses the problem of finding optimum routes or paths through networks or grids. A typical metaphor is the problem of how a tourist walking Manhattan

could visit the most attractions with the least walking. Given the grid-like street layout, there are many different paths that could be taken to include MoMA, the Empire State Building, Times Square, and Wall St. (Another version of the problem, a slightly older metaphor, is the travelling salesman problem. How can a travelling salesman visit all the towns in the region doing the least driving?)

Many different situations can be translated into the model of the Manhattan tourist itinerary, even non-spatial ones. When a bioinformatics textbook claims 'development of new sequence comparison algorithms often amounts to building an appropriate "Manhattan"', it emphasises the intimate connection between space (data structures) and itineraries or paths (algorithms). The movement of tourists visiting attractions dotted on the grid of Manhattan's streets provides a useful spatial model or 'data structure' for other forms of movement. So too, the Viterbi algorithm maps sequences of events, the possible outputs of the Markov model onto a grid. Dynamic programming is then used to find optimum path through the grid, which in turn translates into the most probable sequence of states in the model that could have yielded that output. Logistical problems concerning networks, ranging from flight scheduling to text searching, make use of dynamic programming approaches. Dynamic programming models all problems in terms of traversals of directed acyclic graphs (DAGs), or networks in which movement can never go in circuits. By transforming problems such as sequence comparison into a graph/network data-structure, and carefully planning the order in which computations are carried out, dynamical programming algorithms drastically reduce the time needed to find the optimal itinerary.

The dynamic programming technique has been imitated and varied in applications across bioinformatics, logistics and telecommunications because it lays down an especially effective concatenation of steps between source and destination. No technologically determined acceleration in hardware or processor speed is at stake in this movement. Rather, by virtue of a reversal that lies at the heart of dynamic programming algorithms, the Viterbi algorithm traverses the distance between input and output counter-intuitively. That is, it carries out calculations in a different order to that dictated by a practical sense of the world as something to be acted on effectively. They undergo topological transformations and re-ordering in time. The transformations in computational time wrought by dynamic programming proceeds via a process of

intensification of movement that needs to be explained rather than simply attributed to algorithms or to abstraction *per se*. The hack at the centre of 'the creative production of abstraction' in the Viterbi algorithm is not based on a better representation of something, for instance, of the prototypical situation modelled in software. Nor is it based on a solution that puts an end to movement.

In finding the shortest path through a network, the problem is to minimise repeat visits to nodes. In this sense, the algorithm seeks to reduce repetition not to multiply it. Dynamic programming is a tactic to avoid doing calculations that have already been performed or returning to points in the network that have already been visited. Every repeated calculation equates to the tourist backtracking over the same ground. In order to minimise repetition, dynamic programming abstracts from the particular to the general. Rather than trying to solve the specific problem of how to get from one point to another most directly (for example, the minimum number of blocks to walk to visit all the popular sights), dynamic programming algorithms build a table or array containing the best scores for all possible movements (for instance, between all known tourist destinations in Manhattan). A new, conventional order, that of the table or matrix, replaces the topology of the network or the topography of the city. Movements in the order space can be carried out without repetition because the table represents the cost of different movements. Expressed in the most elementary form, by solving the general problem of movements between every point, dynamic programming makes finding the best movement between two chosen points much more efficient. Somewhat paradoxically, by tackling all problems to start with, it accelerates solving a particular one. The creative 'hack' is to reverse the intuitive approach that would start with the particular and then generalise from it. Dynamic programming imagines all movements in order to choose one.

Mental objects, repetition and concatenated worlds

Where have we moved in relation to algorithms as ideal distillations of machine time? Solitary machine time is replaced by proximities between sights, and by an implication of different kinds of action with each other. Three general features emerge from the Viterbi algorithm. Firstly, an algorithm differs from 'brute force' or bare mechanical repetition by virtue of some abstraction that hacks, twists or fold the space and time of computation. For instance, the Viterbi algorithm accepts 'hidden states' as a given, not

as something to be excluded. The abstract ideal of bare repetition does not capture the pragmatic reality of algorithms. To treat algorithms as pure repetition would be to overlook the inventive variation embodied in every algorithm. It accepts a certain non-ideality as given. An algorithm often turns uncertainty into a feature that allows repetition to gain traction in situations where conditions are far from ideal, where evolutionary history or abundant communication generates constant variations and differences. In this sense, it moves to the fringes of repetition.

Secondly, the treatment of repetition found in the algorithm is variable. It is not pure repetition, but a mixture of different repetitions that intensify movement rather than simply automate it. For instance, the Viterbi algorithm assumes that observed events occur in a sequence. It also assumes that every observed event corresponds to one hidden event. Having made idealised assumptions about the existence of events in sequence, the algorithm starts rearranging that sequence in order to move through it differently, and to associate things in the sequence together. For instance, as mentioned above, dynamic programming algorithms make all possible moves in advance and then combine them in synthetic forms to quickly find the best move. Although repetition is ordered in algorithms, it is modulated and varied in ways that augment difference rather than simply reducing it to the same.

Thirdly, through algorithms, variation in repetition become associative and concatenate different relations. Algorithms seems to distil pure repetition. They seem to be one of the most complete forms of repetition. After all, the very notion of an algorithm is of an ordered sequence of operations that can be carried out by anyone or by anything. How then could it be said that an algorithm enlarges the 'frame of the now'? As much social theory argued over the last century (at least beginning with Nietzsche), pure or 'bare' repetition is a normative ideal that serves to stabilise power relations and social forms. Repetition is at best a provisional accomplishment, not a given, amidst variation. Actually, repetition constantly varies and differs, as it is imitated. This variation itself may be imitated, and in an important sense, this is what algorithms offer - something made to be imitated. (For instance, as mentioned earlier, the Viterbi algorithm has been imitated and varied in several hundred patented versions.) The practical function of computer science courses, textbooks on algorithms, software libraries and VLSI chips is to make treatments of repetition imitable. Algorithms socialize repetition by making

it imitable. The force of an algorithm in establishing repetition is dependent on the linkages it establishes, the predicates or attributes it brings into relation, and corporealises as a force with specific intensity and quality.

These features allow us to begin understanding algorithms from the perspective of force and desire, and to fit living invention and repetition together. William James wrote:

Two parts, themselves disjoined, may nevertheless hang together by intermediaries with which they are severally connected, and the whole world may hang together similarly, inasmuch as *some* path of conjunctive translation by which to pass from one of its parts to another may always be discernible.

Following James' logic of conjunction, we could say that algorithms function as associative objects. Algorithms make the world they work in hang together in certain ways and not others today. They give weight to relations, and they treat relations as real by holding things together, and by making some conjoined path of translation discernable.

Conclusions: machine time as practice of relationality

Any technology confronts the realisation that no foundation can fully secure its operations. This exploration of the Viterbi algorithm is motivated by the idea that technical infrastructures and operations are not simply mechanical foundations of forms of life and sociality. Their appearance as infrastructure or as machine time is relative to the irreducible traces of embodiment and sociality that they carry with them. These traces are legible at the cost of some willingness to regard technical arrangements as analysable practices of relationality. Algorithms are 'global phenomena' in a specific sense proposed recently by Stephen Collier and Aihwa Ong:

Global phenomena are not unrelated to social and cultural problems. But they have a distinctive capacity for decontextualisation and recontextualization, abstractability and movement, across diverse social and cultural situations and spheres of life. ... At the same time, the conditions of possibility of this movement are complex. Global forms are limited or delimited by specific technical infrastructures, administrative apparatuses, or value regimes, not by the vagaries of a social or cultural field.

In their orderings of computation and communicational times and spaces, algorithms such as Viterbi demonstrate specific kinds of decontextualisation. The combination of convolutional codes and Viterbi decoding contributes to the possibility of billions radio signals propagating at the same time in urban environments. However, the overall argument in this chapter has concentrated on suggesting that this decontextualising

effect only succeeds in organising machine time by folding different relations together. Two significant temporal restructurings result from this folding together of relations. The incorporation of past states into each message begins to break down the punctual, discrete nature of repetition. A flow of information begins to look more like a phenomenological 'now', composed of retentions and protentions. The treatment of itineraries by putting the general before the specific, by solving all problems in order to solve the particular, treats movement in space as a crowd phenomena, not as an solitary movement.

The argument that machine time is relational and heterogeneous is both an empirical claim concerning the actual state of communications today, and a more general claim about how technology can be thought today. The value of examining a particular algorithm is showing how complex the patchwork of orderings, materialities, and social orderings of communications can be. Unpacking a particular algorithm also begins to lend historical depth to machine time. The mixture of statistics, logistics, information theory and signal processing conventions present in the Viterbi algorithm points to several centuries of governmentality, sciences, commerce, organisational practice and management. These diverse materials are sedimented not just in perception, but in things. Because they increasingly rely on hidden states, algorithms construct microworlds from concatenations, collections and paths that it brings into mosaic conjunction. These microworlds possess qualities that change our experience of force, movement, feeling and duration.

References

Bergson, Henri Louis, and Arthur Mitchell. *Creative Evolution*. New York,: H. Holt and company, 1911.

Cubitt, Sean. *Digital Aesthetics*. London ; Thousand Oaks, Calif.: SAGE, 1998.

Derrida, Jacques. *Edmund Husserl's Origin of Geometry, an Introduction*. Lincoln: University of Nebraska Press, 1989.

Gell, Alfred. *The Anthropology of Time : Cultural Constructions of Temporal Maps and Images, Explorations in Anthropology*. Oxford ; Providence: Berg, 1992.

Graham, Paul. *Hackers & Painters. Big Ideas from the Computer Age*. Sebastapol, CA: O'Reilly, 2004.

Hansen, Mark B. N. *New Philosophy for New Media*. Cambridge, MA & London: The MIT Press, 2004.

Husserl, Edmund, and John B. Brough. *On the Phenomenology of the Consciousness of Internal Time (1893-1917)*. Dordrecht ; Boston: Kluwer Academic Publishers, 1991.

IEEE. "IEEE Std 802.11b-1999 Part 11: Wireless Lan Medium Access Control (Mac) and Physical Layer (Phy) Specifications: Higher-Speed Physical Layer Extension in the 2.4 Ghz Band." New York: Institute of Electrical and Electronics Engineers, Inc., 1999.

James, William, and Ralph Barton Perry. *Essays in Radical Empiricism*. New York: Longmans, Green, and co., 1912.

Jones, Neil C., and Pavel Pevzner. *An Introduction to Bioinformatics Algorithms, Computational Molecular Biology*. Cambridge, MA: MIT Press, 2004.

Kittler, Friedrich. "There Is No Software." In *Literature Media Information Systems*, edited by John Johnston. London: G&B Arts International, 1997.

Knorr-Cetina, Karin, and Urs Bruegger. "The Market as an Object of Attachment: Exploring Postsocial Relations in Financial Markets." *Canadian Journal of Sociology* 25, no. 2 (2000): 141-68.

Lesk, Arthur M. *Introduction to Bioinformatics*. Oxford ; New York: Oxford University Press, 2002.

Levitin, Anany. *Introduction to the Design and Analysis of Algorithms*. Boston, MA: Addison Wesley, 2003.

Lunenfeld, Peter. *Snap to Grid : A User's Guide to Digital Arts, Media, and Cultures*. Cambridge, Mass.: MIT, 2000.

MacKenzie, Donald A. *Knowing Machines : Essays on Technical Change, Inside Technology*. Cambridge, Mass. ; London: MIT Press, 1996.

Manovich, Lev. "The Labor of Perception." In *Clicking In : Hot Links to a Digital Culture*, edited by Lynn Hershman-Leeson, 183-93. Seattle: Bay Press, 1996.

———. *The Language of New Media*. Cambridge, MA: MIT Press, 2001.

Oliver, Ian. *Programming Classics : Implementing the World's Best Algorithms*. New York: Prentice Hall, 1993.

Ong, Aihwa, and Stephen J. Collier. *Global Assemblages : Technology, Politics, and Ethics as Anthropological Problems*. Malden, MA: Blackwell Publishing, 2005.

Probyn, Elspeth. "Anxious Proximities." In *Timespace : Geographies of Temporality Critical Geographies ; 13*, edited by Jon May and Nigel Thrift, 171-84. London: Routledge, 2001.

Qualcomm. *About Qualcomm*, 2005 [cited 4 May 2005]. Available from <http://www.qualcomm.com/>.

Rabinow, Paul. "Midst Anthropology's Problems." In *Global Assemblages : Technology, Politics, and Ethics as Anthropological Problems*, edited by Aihwa Ong and Stephen J. Collier, 40-54. Malden, MA: Blackwell Publishing, 2005.

Raymond, Eric. *The New Hacker's Dictionary*. Cambridge, MA: MIT Press, 1996.

Rodowick, David Norman. *Reading the Figural, or, Philosophy after the New Media*. Durham: Duke University Press, 2001.

Stiegler, Bernard. *La Technique Et Le Temps, La Philosophie En Effet*,. Paris: Galil ee/Cit e des sciences et de l'industrie, 1994.

———. *La Technique Et Le Temps. 3, Le Temps Du Cin ema Et La Question Du Mal- etre, La Philosophie En Effet*,. Paris: Galil ee, 2001.

Technology, National Institute of Standards and. "Big-O Notation." (2005).

Thrift, Nigel. "Movement-Space: The Changing Domain of Thinking Resulting from the Development of New Kinds of Spatial Awareness." *Economy and Society* 33, no. 4 (2004): 582-604.

———. "Remembering the Technological Unconscious by Foregrounding Knowledges of Position." *Environment & Planning D: Society & Space* 22, no. 1 (2004): 175-91.

US Patent Office. *Viterbi Decode Patents*, 2005 [cited 4 May 2005]. Available from <http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=%2Fnethtml%2Fsearch-adv.htm&r=0&f=S&l=50&d=PTXT&RS=TTL%2Fviterbi&Refine=Refine+Search&Refine=Refine+Search&Query=TTL%2Fviterbi>.

Virilio, Paul. *Open Sky*. London: Verso, 1997.

———. "The Perspective of Real Time." In *Open Sky*, 22-34. London: Verso, 1997.

Virno, Paolo. *A Grammar of the Multitude : For an Analysis of Contemporary Forms of Life, Semiotext(E) Foreign Agents Series*. Los Angeles: Semiotext(e), 2004.

Viterbi, Andrew J. " Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm." *IEEE Transactions on Information Theory* 13, no. 2 (1967): 260-67.

Wark, McKenzie. *A Hacker Manifesto*. Cambridge, MA: Harvard University Press, 2004.

Wikipedia. *Viterbi Algorithm*, 2005 [cited 4 May 2005]. Available from http://en.wikipedia.org/wiki/Viterbi_algorithm.

Wirth, Niklaus. *Algorithms + Data Structures=Programs*. Englewood Cliffs, N.J.: Prentice-Hall, 1976.

Wolfram, Stephen. *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002.