Algorhythmics: Understanding Micro-Temporality in Computational Cultures

2012-09-12 11:09:38 andrew

Abstract
While in the terminology of the computational sciences an algorithm is often defined as a finite sequence of step-by-step instructions, which "bear a crucial, if problematic, relationship to material reality,"[1] rhythm, a term closer to the study of cultural phenomena, shall be defined as an elementary movement of matter, bodies and signals, which oscillate in-between the discrete and the continuous, between the symbolic and the real, between digital and analogue. This article considers the specific role of algorithms and their rhythms. It not only addresses some important historical dimensions of contemporary computational culture, but also analyses algorithms from a systematic point of view, specifically in relation to software-induced breakdowns of distributed networks.

Introduction
Algorithms are mathematical and thus abstract structures, but should not be mistaken for algebraic formulae, since assignments or instructions operated by algorithms are non-reversible. They are vector-dependent and have built-in time functions. Their ties to machinic reality and operability make algorithms time-based and as such part of rhythmic procedures, which are able to cause measurable temporal effects. The interleaving of both concepts – algorithm and rhythm – is here termed algo*rhythm* and was formerly a heuristic word play, but has become a critical, media archaeological concept, revealing media epistemic aspects of current computational culture and its history. An algo*rhythm* is the result of an inter-play, orchestration and synthesis of abstract algorithmic and calculable organisational concepts, with rhythmic real-world signals, which have measurable physical properties. To understand the implementation of algo*rhythmics*, it is necessary to cultivate a specific focus on micro-temporal and molecular zones of the mediated technical environments created and generated by time-critical processes[2] and the media operability of information storage, transmission and processing.

A very short recursive cut into the 12th and 13th centuries permits the detection of the first strata of the concept of algorithm in medieval Europe. Later, the appearance of the term algorithm was part of computer scientific jargon and the emergence of software in the early 1960s. Of particular interest is the widespread engineering routine of listening to the algo*rhythmics* of computational processes by using unsophisticated auditory interfaces, which was put into practice between 1949 and 1964. With the dawn of software-based operating systems this routine became redundant and disappeared. My argument in this article is that despite this, the use of algo*rhythmics* as a method of critical enquiry into computational culture and their operability is applicable to understanding later periods of the history of computational culture. My article affirms the versatility of algo*rhythmics,* especially in combination with the notion of *agencement* (explained below), through a consideration of the crash of the long-distance telephone network in North America in January 1990 (the "AT&T-Crash").

It is my final argument that the concept of algo*rhythmics* not only sheds new light on the workings of early mainframe computers, but explains some essential parts of the distributed dynamics of contemporary digital networks within the "technological unconscious"[3] in the early 21th century and might thus be helpful for analysis of current computational cultures.

Deep recursion. Algorism 1350
The term 'algorithm' emerged in late medieval Europe, and more precisely in Spain during the 12th century while the scripts of the Arabian mathematician Muḥammad ibn Mūsā al-Khwārizmī were translated into Latin. The only sources available today are not in Arabic, but in Latin and often begin with the phrase "Dixit Algorismi", which means "Algorismi said" and refer to Al-Khwārizmī.[4] These scripts describe the method of addition, subtraction, multiplication and division with the Hindu-Arabic numeral system, which was strikingly new; in the ancient Greek and Roman numeral systems there was no concept of the number zero as a placeholder for powers of ten. The ancient Greeks calculated mostly with shapes, lines, surfaces and things, but not with numbers. Geometry was more important than arithmetic.[5] For a long time "algorism" meant the specific step-by-step method of performing written elementary arithmetic inscribed on materials such as sand, wax and paper[6] using operations such as addition, deletion and shifting of numbers in the specific positional system of Hindu-Arabic numerals.[7] In this sense the concept of algorism was revolutionary and the basic operations of later computing processors are still derived from the positional system not of ten states, but of two (0 or 1), which are then shifted, added, deleted and moved. The underlying principles of symbolical operation are, in an abstract sense, still the same.[8]

Popularization of algorithmic thinking 1960
An algorithm formulated in a programming language is not the same as a list of algebraic formulae. Mathematicians of the 20th century such as Alonzo Church (1903–1995) or Stephen C. Kleene (1909–1994) already used the term algorithm in the 1930s and 40s, whereas Alan Turing (1912–54) or Kurt Gödel (1906–1978) scarcely used it. With the dawn of higher level programming languages such as Algol 58, Algol 60 and all the other Algol inspired languages in the early 1960s, the notion of algorithmic notation spread throughout the academic world of scientific computation. Algol is the abbreviation for ALGOrithmic Language. It was the result of a pan-Atlantic collaboration between many different programming pioneers and mathematicians such as Friedrich L. Bauer (*1924), Hermann Bottenbruch, Heinz Rutishauser (1918–1970) and Klaus Samelson (1918–1980), who were members of the European group and John W. Backus (1924–2007), Charles Katz, Alan J. Perlis (1922–1990) and Joseph H. Wegstein from the US. The later Algol 60 language was created with the additional work of John McCarthy (1927–2011) and Peter Naur (*1928).[9]

The North American pioneers proposed that Algol 58 should be named *International Algebraic Language*, which shows their lack of conceptual clarity concerning the differences between the meaning of algorithmic and algebraic. The Europeans must have protested vehemently against the idea of calling their new language algebraic, since the term algorithm was more established in their emerging community of scientific calculation. October 1955 saw the Pan-European conference in Darmstadt Germany, with the title "Elektronische Rechenmaschinen und Informationsverarbeitung," where pioneers such as Herman H. Goldstine (1913–2004), Andrew D. Booth (1918–2009), Howard H. Aiken (1900-1973), Konrad Zuse (1910-1995), Friedrich L. Bauer (*1924) and Edsger W. Dijkstra (1930-2002) presented their work. It was at this conference that Rutishauser presented for the first time his concept of an algorithmic notation system to a broader audience. Thereby he clearly distinguished it from algebraic notation systems.

> "The equal sign is substituted by the so called results-in sign ⇒ by K. Zuse, which demands, that the left values shall be calculated as declared into a new value and shall be named as declared on the right side (in contrast to a + b = c, which in sense of the algorithmic notation would merely be a statement)."[10]

Donald E. Knuth (*1938) himself a pioneer in algorithmic thinking of a slightly younger generation and a software historian too wrote:

“[M]athematicians had never used such an operator before; in fact, the systematic use of assignments constitutes a distinct break between computer-science thinking and mathematical thinking.”[11] It can be argued that this important shift in the mode of understanding was induced by the machinic reality of computers and their boundedness to time-based processes. While in algebraic notation a + b = c is merely a symbolic-abstract statement which is not bound to any form of concrete execution, and thus can be changed in to a = c – b or b = c – a. The execution of the assignment a + b ⇒ c is non reversible. Only once the values of a + b have been transferred and assigned to c can it get overwritten, reassigned to another variable or deleted. To convince the Americans of the theoretical importance of this distinction must have been one of the aims of the European fraction of the Algol team, but this change seemed to be quite difficult. John Backus didn’t change his vocabulary even after the first meeting of 1958 in Zurich, Switzerland. In a 1959 document called *The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference* he still wrote algebraic and never used algorithmic. Only the fast dissemination of Algol 60, especially in the emerging computer science departments in Europe, helped to establish the new way of algorithmic thinking.[12]

Computational culture of algo*rhythmics* 1949-1964

Before the popularization of the concept of algorithm in the early 1960s, there was an almost forgotten computational culture of machine listening. Alongside the first visual interfaces to early mainframe computers – such as the Williams-Kilburn Tube operating for the first time in 1948 within the circuits of the Manchester Small-Scale Experimental Machine (SSEM) – or the many type-machine-like printing outputs of standard calculating machines, there were also *auditory interfaces*, which were simple amplifier-speaker systems built-in to the circuits of the early mainframe computers. Examples of such machines were the famous UNIVAC I, the TX-0 at MIT, the CSIRAC in Australia and the Pilot ACE in England, as well later machines such as the Pegasus produced by Ferranti Ltd. in Manchester and the PASCAL-Computer of Philips Electronics in Eindhoven, Netherlands.

Audio example 1, excerpt from side B/ track 1 on vinyl recording accompanying Nijenhuis, “Listening in to the PASCAL”.

There is an audio recording left behind from the computational culture of the latter machine,[13] which can be analysed epistemologically using media archaeological methods.[14] The following is an in-depth analysis and excavation focusing on a specific recording of the sound of the machine calculating whether or not some large numbers (with ten zeros or more) are prime numbers. An excerpt is audible as audio example 1. Finding out if a number is a prime number or not, is a matter of division, comparison, counting, shifting and adding. Similar calculating operations had already been described in principle in the *algorismi* manuscripts of the 12th and 13th centuries. Fig. 1 shows the algorithm and flowchart that the engineers and programmers of the PASCAL-Computer used.

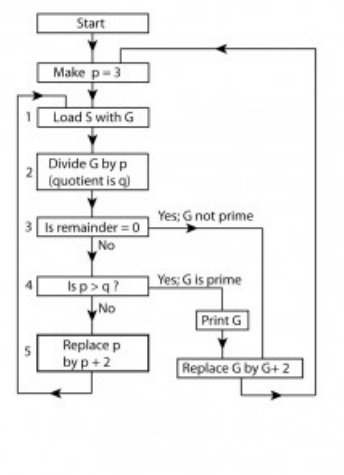

Fig. 1, flow chart of computing program used in the search for prime number as represented in Nijenhuis, “Listening in to the PASCAL,” 167, Fig. 3.

A concrete example will exemplify the algorithm. The number for which primeness is to be determined shall be 443. As defined by the algorithm the divisor p is 3. The value 443, which is stored in G gets transferred to shift-register S, which is operated by step 1. Step 2 is performing the division. The first quotient q would then be 147. 3 times 147 is 441. 443 minus 441 is 2, which is the remainder. In step 3 we check, if the remainder is zero and go to step 4 if no. Here we check if p, which is 3, is bigger than q, which is 147. The answer is no, so we go on to step 5, add the number 2 to p and go back to step 1. P is now 5 and after the division the quotient q is 88. In the following we would loop step 1 to 5 altogether additional 9 times, until the divisor p gets bigger than the quotient q and we could prove that 443 is a prime number. The series of quotients created is stored in shift-register S. In the present case it would consists of the numbers 88, 63, 49, 40, 34, 29, 26, 23, 21, 19. If you look only at whether the numbers are odd or even you can see that there is some rhythm of change between the two properties, which can by expressed as E, O, O, E, E, O, E, O, O, O, while E stands for even and O for odd.

The technical area within the electronic circuit and ensemble of the PASCAL, where some key operations of the algorithm were transformed into audible algo*rhythms* was the last flip-flop of shift-register S. The electronic signals at this passage were amplified by a simple loudspeaker-amplifier set-up and transformed into mechanical movement of the loudspeaker’s membrane and therefore sound. Fig. 2 represents the electronic signal and changes of potential, which is measurable during one iteration of the above-described algorithm for finding out if a number is a prime number or not.

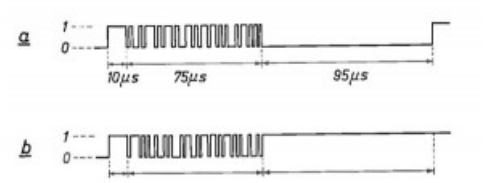

Fig. 4. Voltage variations on the loudspeaker during the thickly drawn cycle in fig. 3.
a) Quotient q even, b) q odd.

Fig. 2, electronic signals as represented in Nijenhuis, “Listening in to the PASCAL,” 168, Fig. 4.

A LOW or almost no signal represents a symbolic 0 in shift-register S and at the same time it means that there is an even number stored in it. This is represented by the waveform a at the top of the diagram. As seen in waveform b at the bottom, if you have a 1, then you have a HIGH, it means an odd number. This number is the quotient q and thus an important intermediate result of Step 2 as shown on the flow chart of the algorithm above. As mentioned earlier, during the computation process, the algorithm will produce a variety of different rhythms concerning the pattern of even and odd numbers, thus it will produce different series of waveform a and b in many variations, while the basic cycle of 180 µs is kept. Therefore a 5.55 kHz tone is audible throughout the whole calculation process. At the same time a sonic process, which could be described as howling is audible. It produces some white noise, but then gets continuously slower, until you can hear the up and down of the frequency very clearly. This happens especially with very big numbers, because the bigger the number, the longer the series of quotients and the more regular the rhythm of changes between odd and even numbers.[15] The howling is created by a process called frequency modulation, which can be understood as the coupling of two oscillators. In this manner many different frequencies can be produced. And since the change of rhythms in the alternation of odd and even-quotients happens gradually, the algorithm produces the specific kind of howling audible on the vinyl recording from 1962 made by the engineers from Philips.

The earliest historical evidence for the engineering routine of algo*rhythmic* listening is so far a statement by Louis D. Wilson, one of the main engineers of the BINAC and the UNIVAC-I. In 1990 at the so called UNIVAC-Conference organized by the Charles Babbage Institute of the University of Minnesota, Wilson recalled how the technical listening method evolved around the year 1949 as a practical procedure to analyse the computer:

> “When we were testing BINAC out, we were working two shifts and we worked all night long, and we had a radio going. After a while we noticed that you could recognize the pattern of what was happening at the moment by listening to the static on the radio. So I installed a detector in the console and an amplifier and a speaker so that you could deliberately listen to these things.”[16]

Christopher P. Burton (*1932), who was from 1957 to 1989 an engineer of the Ferranti Ltd. computing department in Manchester, co-founder of the Computer Conservation Society in England and the chief engineer of the *Manchester Small-Scale Experimental Machine-Resurrection-Project* wrote in an email interview with the author:

> “[O]n Pegasus, the test programs are permanently stored on an isolated part of the magnetic drum, so a few simple manipulations of the control switches could call whatever test program was currently of interest. A working program had its characteristic sound (by chance, depending where the noise probe was connected) and the sound just changed when a fault was detected.”

According to Burton, the position of the so-called “noise probe” was variable, thus different sound sources could be heard and auscultated. These could be individual flip-flops in different registers, different data bus nodes or other streams and passages of data traffic. Not only was passive listening of the computer-operations very common, but was also an active exploration of the machine, listening to its rhythms. Burton continues,

> “Very common were contact failures where a plug-in package connector to the ‘backplane’ had a poor connection. These would be detected by running the Test Program set to continue running despite failures (not always possible, of course), and then listening to the rhythmic sound while going round the machine tapping the hardware with the fingers or with a tool, waiting for the vibration to cause the fault and thus change the rhythm of the ‘tune’.”

The routine of algo*rhythmic* listening was widespread, but quickly forgotten. One convincing reason for the lack of technical terms such as algo*rhythm* or algo*rhythmic* listening is the fact that the term algorithm itself was not popular until the 1960s. Additionally, in the early 1960s many chief operators and engineers of mainframe computers were made redundant, since more reliable software-based operators called *operating systems* were introduced.[17] These software systems were able to listen to their own operations and algo*rhythms* by themselves without any human intervention. Scientific standards and rules of objectivity might have played an important role as well, since listening was more an implicit form of knowledge than any broadly accepted scientific method.[18] Diagrams, written data or photographic evidence were far more convincing in such contexts. Finally, the term rhythm itself is rarely used in the technical jargon[19] – instead the term ‘clock’ or ‘pulse’ is preferred.

Algo*rhythmic* studies as software studies

The concept of algo*rhythm* in the context of software studies and media archaeologies of computational cultures is valuable regardless of its non-appearance in the technical jargon of computational sciences, because it allows for a more process-based approach, which goes beyond the limits of software studies. According to Matthew Kirschenbaum there is always a gap between the deepest level of software-based analytics of the materiality of digital storage and its deepest signal and hardware-based level.[20] An algo*rhythmic* study cultivates a close reading of, or to be more accurate, a close *listening to* the technical workings of computers and their networks on precisely this deep level of physical signals. Analysed on different scales from the microprogramming on the level of the CPU, or the physical layer of communication protocols between server and clients, to interface applications on all levels where human interaction is

needed, to longer time spans such as the daily backing up of large amounts of data, distributed and networked algo*rhythmic* processes can be found on many levels of our current informational networks and computational cultures. These are more and more connected via wireless technologies as well.[21]

The notion of algo*rhythm* points to the very processes of computation itself and this is its epistemic quality. It manifest itself as an epistemic model of a machine that makes time itself logically controllable and, while operating, produces measurable time effects and rhythms. Such machines as the mechanical clock or the music automaton were introduced to Europe in the 1300s, around the same time as the notion of algorithm – as algorism. The first arithmetic calculation machine invented 1623 by Wilhelm Schickard (1592-1635) was called a calculating clock (Rechenuhr). In the next step Charles Babbage (1791-1871) called his calculator *engine*, which emphasized its transformative power and its ability to make recursions. In 1936 Alan Turing (1912-1954) wrote a paper about *machines*, specifically one machine which can simulate all other machines. More than forty years later, Gilles Deleuze and Félix Guattari catalysed the usage of the term *abstract machine* within the context of philosophy and post-structuralism – this after the dawn of software[22] culture and the concept of algorithm as shown above.[23] Since the emergence of Arpanet around 1970,[24] computers not only work alone, but collaboratively and are orchestrated via distributed networks. This was reflected as well by Deleuze and Guattari in their philosophy on the basis of the term “agencement” and their usage of the term “ritornello” (generally translated into English as “refrain”). A consequence of this conceptual trajectory and archaeological short-circuit of more than seven hundred years would be the combination of algo*rhythm* with agencement. Before explaining this, the latter term itself is to be defined.

Agencement not assemblage nor actor-network

The French term “agencement” derived from the verb “agencer”, which means to arrange or to fit together. An agencement is thus an arrangement, assemblage or a laid-out configuration. In ordinary French language it often has a material meaning such as the parts of a machine. Its usual mistranslation into “assemblage” loses its semantics of agency and its sense of actively evoking processes.[25] Michael Callon, who co-invented Actor-Network Theory (ANT) with Bruno Latour, prefers the usage of “agencement” to the misleading term “actor-network”.[26] Latour himself withdraws the usage of the latter term, since it implies many wrong stratifications, which tend to get static.[27] By using agencement as an alternative discursive term these issues can be overcome, as the term agencement involves and intermingles both the procedural and the static aspects of an actor-network. All parts of an agencement can include both agents and networks independently, consecutively or simultaneously. As well they can change their status dynamically, in self-organized and sometimes unexpected ways.[28] The term agencement denotes not only the network, but also all the actions that happen in this network. This is especially true for so-called wireless networks, where there is practically no network anymore. The air space is filled with electromagnetic waves and signals which are themselves the network. A space is not a network, but a channel and container for an infinite number of physical connections and the carrier of an infinite number of signals. It builds the medium of specific micro-temporal algo*rhythmic* agencements. In electromagnetic, informational and computational networks there are no connections, and no wires anymore. The wires are constantly recreated symbolically as algo*rhythmic* signals in the time scale of milli- or microseconds with specific logical signatures or, to be more metaphorical, signals with specific rhythms, which are related again to other machinic rhythms.

Algo*rhythmic* agencement of the AT&T Crash

The advantages of the term agencement over the concept of actor- or agent-network especially in combination with the notion of algo*rhythm* as suggested above is best illustrated by an analysis of the famous[29] AT&T Crash of 1990.

On January 15th in 1990 the long-distance telephone network in North America crashed and was disabled for nine hours. The reason for the crash, discovered later, was a small programming mistake implemented via a software update of the newer telephone line switching centres. The update was made in December 1989 and the whole agencement worked well until one of the 114 switching-computers which were affected started to detect some malfunctions in its system and shut itself down for four to six seconds. This had to be communicated to the other neighbouring switching-computers in order to update their routing-maps, so that the machine that was shut-down could be avoided, because it was not working. The crucial point is that the malfunction became effective and detected as a result of this process of updating the virtual map of all neighbouring switching-computers.[30] Each computer, in turn updating its routing map, shut itself down. And consequently the neighbouring computers all shut themselves down, and so on. The whole network and its agencement was trapped in an eternal algo*rhythmic,* distributed and polyrhythmic refrain.

The cause of the malfunction was a simple programming mistake in the programming language C. In a long “do-while-loop”, there were many switch-instructions and in one of the many branches a switch-clause had an “if-loop” in it, which should have been fine, but additionally there was a break-instruction in it, which in the programming language C doesn’t make a lot of sense.[31] The program did not act in the intended way and the rhythms concerning its computational operations were thus out of order. The break-outs of loops or the breaks of switch instruction chains were timed in unintended rhythms. These evil rhythms were usually not detectable by the self-monitoring operating systems of the switching machines, but became effective during the already-mentioned updating process of the routing maps. Two levels of rhythmicity constituted the specific algo*rhythmic* interplay and machinic over-functioning, which was distributed over the network of the switching-computers via the agencement of software messages and malfunctions that caused the crash. On one side there were the rhythms happening within one switching-computer itself, but then on the other side was the rhythm and timing of the notification messages, which were sent out from one switching computer to all its neighbouring computers. Receiving this message, these neighbouring computers crashed as well, but then after a short break of four to six seconds,[32] sent messages to their neighbouring computers again. In this way the crashing and shutdown of the switching-computers spread rhythmically and cascaded over the whole long-distance telephone network in North America and repeated the overall process for nearly nine hours.

An engineer from AT&T, who was working on these problem remembers: “[W]e knew very early on we had a problem we’d never seen before. At the same time, we were looking at the pattern of error messages and trying to understand what they were telling us about this condition.”[33] The aforementioned programming mistakes determined the emergent behaviour of the algo*rhythmic* agencement within the AT&T-long-distance telephone network. Or to describe it more poetically: The ghost in the machine became active and the system behaved in unexpected ways. It became a ‘bad’ system.[34] To understand the emergent behaviour, which led to the crash, the engineers looked at the rhythm and pattern of system messages. These operated on a higher level than the algo*rhythms* on the physical-layer of protocols, since listening to algo*rhythmics* was not common practice anymore. Only a media archaeological approach, in combination with a telephonic enquiry into the agencement of the AT&T-long-distance telephone network, as a method of understanding its algo*rhythmics,* could excavate such historical and temporal details of software history and the events of computational culture such as the one described above.

Understanding the algo*rhythmics* of computational cultures

Specific bad situations within certain algo*rhythmic* agencements such as the AT&T Crash can cause epistemic effects, which can provoke in-depth analytics initially on the part of responsible engineers and technicians, and later by all kinds of interested people such as hobbyists, geeks, criminologists, security-related researchers, media archaeologists and scholars of contemporary computational cultures. Cases of such specific studies could be Black Monday in 1987, the Flash Crash of May 6th 2010, the software-induced accident by Amazon.com selling a book about flies for 24 Million Dollars in April 2011, the many Bitcoin crashes during 2011 and countless other examples of crashes, breakdowns and other accidents, where the performance of algo*rhythmic* agencements played constitutive roles.

To conclude and in recursion to the beginning of the article it is important to emphasize again the time-boundedness of computational culture. It is common sense that on a micro-level of critical enquiry algorithms are crucial elements in any software culture. They are the basis of our current technological everyday lives and its infrastructure. The article argued that there is a rhythmicity of computation and communication. Its speeds and spatio-temporal agencement depends on the historical and technological contexts of the concrete machines and protocols, which may vary from minutes, to seconds, milliseconds and microseconds. Since the dawn of Hindu-Arabic arithmetic, shifting, deleting, and adding positional numerals became the principle operations of calculation and later, of computational machines. The principal technicity[35] of these Arabic "algorisms" as operations of shifting, deleting, adding, is evidenced in the ⇒ sign by Konrad Zuse and was later incorporated in the syntax of higher level programming languages such as Algol 60, as well as inscribed on the level of microprogramming in today's CPU. A contemporary algo*rhythmic* agencement is still based on the timed and controlled sending, receiving, storing and processing of physical signals – operations which have their specific time effects and rhythms. The two detailed case studies, the first of the listening practice in the context of early mainframe computing, and the second of the agencement of telephone networks suggested that the analysis of their algo*rhythmics* is crucial for the understanding of their technical workings and their specific rhythmic, timed and protocol-driven orchestration.

A study of the epistemic, social, political and cultural effects of computational processes and their effects uses software and other media to listen, observe, analyse and synthesize them. Following the methods of the so called 'digital humanities' an algo*rhythmic* study of computational cultures will need to program their own software.[36] As such, understanding computation means doing epistemic reverse-engineering of their inbound and outbound processes, signals and rhythms while not concentrating too much solely on the visual aspects of the agencement of enquiry, but also on its audible, tactile, vibrational, more dynamic and ephemeral aspects. While doing that, the question of the tangibility of the relevant signals, electromagnetic effects, rhythms and machinic rumblings of such networks comes into focus, since many of the processes are at work on micro levels. Nevertheless little effort is needed to make such phenomena accessible to the human senses. In this sense I suggest the use of the term trans-sonic, which denotes the realm of all signals, oscillations, rhythms and flickerings[37] that are inaudible for humans, but can be made – to a certain degree with certain losses – audible again by the use of media technologies such as audio amplification, radio demodulation or software sonification. These hardware-based methods as well as software-based symbolic and mathematical methods of data mining and aestheticization become important for such studies. The range of the sonic results can vary from dense noise, to more identifiable frequencies and melodies, to pulsating rhythms. These aesthetic qualities are not to be mistaken for epistemic qualities, as the specific way that the algo*rhythms* will sound will depend on the decisions of the investigators. Furthermore the sonic results would benefit from being combined with visualizations, diagrams and signs. Due to the universal structure of digital data, it is possible to conduct algo*rhythmic* studies of computational culture not solely with the use of sonic interfaces such as loudspeakers, but also by rhythmicizing eyes and skins. In fact, a balanced and appropriate use of all senses might produce the best results for such an epistemic and media archaeological enquiry.

Bibliography

UNIVAC Conference, OH 200. Oral history on 17-18 May 1990. Smithsonian Institution Washington, D.C., University of Minnesota, Minneapolis: Charles Babbage Institute.

Abbate, Janet. *Inventing the Internet.* Cambridge, MA: MIT Press, 1999.

Bloom, Jonathan M. *Paper before Print. The History and Impact of Paper in the Islamic World.* New Haven/ London: Yale University Press, 2001.

Bullynck, Maarten "Apeiron – Archimedes und die Grenzwerte des griechischen Alphabets," In *Die Geburt des Vokalalphabets aus dem Geist der Poesie. Schrift, Zahl und Ton im Medienverbund,* edited by Wolfgang Ernst and Friedrich Kittler, 199-213. Munich: Fink, 2006.

Callon, Michel. "What does it mean to say that economics is performative ?" In *Do economists make markets? On the performativity of economics,* edited by Donald MacKenzie, Fabian Muniesa and Lucia Siu, 311-354. New Jersey: Princeton Univeristy Press, 2007.

Corbató, Fernando J. et al. *The Compatible Time-Sharing System. A Programmer's Guide.* Cambridge, MA: MIT Press, 1963.

Dantzig, Tobias. *Number. The Language of Science.* Edited by Joseph Mazur. New York: Pi Press, 2005 [1930]

Deleuze, Gilles and Felix Guattari. *A Thousand Plateaus: Capitalism and Schizophrenia.* Translated by Brian Massumi. London/ New York: Continuum, 2004.

Ernst, Wolfgang. "Media Archaeography. Method and Machine versus History and Narrative of Media." In *Media Archaeology. Approaches, Applications, and Implications,* edited by Erkki Huhtamo and Jussi Parikka, 239-255. Berkeley, Calif.: University of California Press, 2011.

Folkerts, Menso and Paul Kunitzsch. *Die älteste lateinische Schrift über das indische Rechnen nach al-Hwarizmi. Edition, Übersetzung und Kommentar.* Munich: C.H. Beck Verlag, 1997.

Goffey, Andrew. "Algorithm." In *Software Studies – a Lexicon*, edited by Matthew Fuller, 15–20. Cambridge, MA: MIT Press, 2008.

Goldschlager, Les and Andrew Lister. *Computer Science. A Modern Introduction.* London: Prentice-Hall, 1982).

Hardie, Iain and Donald MacKenzie. "Assembling an economic actor: the agencement of a Hedge Fund." *The Sociological Review 55/1 Feb.* (2007): 57-80.

Hayles, N. Katherine. *How We Became Posthuman. Virtual Bodies in Cybernetics, Literature, and Informatics.* Chicago: University of Chicago Press, 1999.

Johnston, John. *The Allure of Machinic Life. Cybernetics, Artificial Life, and the New AI.* Cambridge, MA: MIT Press, 2008.

Kirschenbaum, Matthew. *Mechanism. New Media and the Forensic Imagination.* Cambridge, MA: MIT Press, 2008.

Knuth, Donald E. and Luis T. Pardo. "The Early Development of Programming Languages." In *A History of Computing in the Twentieth Century. A collection of essays with introductory essay and indexes, edited* by N. Metropolis, J. Howlett and Gian-Carlo Rota, 197-273. New York: Academic Press, 1980.

Latour, Bruno. "On recalling ANT," in *Actor Network Theory and After,* edited by John Law and John Hassard, 15-25. Oxford: Blackwell, 1999.

Mackenzie, Adrian. *Wirelessness. Radical Empiricism in Network Cultures.* Cambridge, MA: MIT Press, 2010.

Neumann, Peter G.. "Risks to the public in computers and related systems." *ACM SIGSOFT Software Engineering Notes 15/2* April (1991): 3-22.

Nijenhuis, W.. "Listening in to the PASCAL." *Philips Technical Review 24/4-5 (*1962/63): 164-170.

Nofre, David. "Unraveling Algol: US, Europe, and the Creation of a Programming Language," *IEEE Annals of the History of Computing 32/2* (2010): 58-68.

Parikka, Jussi. "Operative Media Archaeology: Wolfgang Ernst's Materialist Media Diagrammatics." *Theory, Culture & Society* 28/5: 52–74.

Parikka, Jussi and Tony D. Sampson. "On Anomalous Objects and Digital Culture. An Introduction." In *The Spam Book. On Viruses, Porn, and Other Anomalies from the Dark Side of Digital Culture,* edited by Jussi Parikka and Tony D. Sampson, 1-18. Cresskill, NJ: Hampton Press, 2009.

Polanyi, Michael. *The Tacit Dimension.* Garden City, NY: Doubleday & Company Inc., 1966.

Phillips, John. "Agencement/Assemblage." *Theory, Culture & Society 23/2-3* (2006):108-109.

Rutishauser, Heinz. "Massnahmen zur Vereinfachung des Programmierens. (Bericht über die 5-jährige Programmierarbeit mit der Z4 gewonnenen Erfahrungen)." *Elektronische Rechenmaschinen und Informationsverarbeitung: Nachrichtentechnische Fachberichte 4* (1956): 26-30.

Rutishauser, Heinz. *Description of ALGOL60.* Berlin: Springer 1967.

Simondon, Gilbert. *Du mode d'existence des object techniques. Thèse complémentaire pour le doctorat és lettres présentée à la Faculté des Lettres de l'Université de Paris.* Paris: Aubier (Edition Montaigne), 1958.

Sterling, Bruce. *The Hacker Crackdown.* New York: Bantam, 1992.

Thrift, Nigel. "Remembering the technological unconscious by foregrounding knowledges of position." *Environment and Planning D: Society and Space 22* (2004): 175-190.

Tukey, John W. "The Teaching of Concrete Mathematics", *The American Mathematical Monthly 65/1* (1958): 1-9.

Williams, Frederic C. and Tom Kilburn. "The University of Manchester Computing Machine." *Proceedings of the Review of Electronic Digital Computers. International Workshop on Managing Requirements Knowledge* (1951): 57-61.

References

1. Andrew Goffey, "Algorithm," in *Software Studies – a Lexicon*, ed. Matthew Fuller (Cambridge, MA: MIT Press, 2008), 16. ↩
2. See Jussi Parikka, "Operative Media Archaeology: Wolfgang Ernst's Materialist Media Diagrammatic," *Theory, Culture & Society* 28/5 (2011): 58. ↩
3. Nigel Thrift, "Remembering the technological unconscious by foregrounding knowledges of position," *Environment and Planning D: Society and Space* 22 (2004): 177. ↩
4. See Tobias Dantzig, *Number. The Language of Science*, ed. Joseph Mazur, New York: Pi Press, 2005 (1930), 33 and Menso Folkerts / Paul Kunitzsch, *Die älteste lateinische Schrift über das indische Rechnen nach al-Hwarizmi. Edition, Übersetzung und Kommentar*, (Munich: C.H. Beck Verlag, 1997). ↩
5. See Maarten Bullynck, "Apeiron – Archimedes und die Grenzwerte des griechischen Alphabets," in *Die Geburt des Vokalalphabets aus dem Geist der Poesie. Schrift, Zahl und Ton im Medienverbund*, ed. Wolfgang Ernst / Friedrich Kittler (Munich: Fink, 2006), 203. ↩
6. See Jonathan M. Bloom, *Paper before Print. The History and Impact of Paper in the Islamic World*, (New Haven/ London: Yale University Press, 2001), 129-133. ↩
7. See Dantzig 2005, 33 and Folkerts/ Kunitzsch 1997. ↩
8. See Les Goldschlager/ Andrew Lister, *Computer Science. A Modern Introduction*, (London: Prentice-Hall, 1982), 136-146. ↩
9. See Heinz Rutishauser, Description of ALGOL60, (Berlin: Springer 1967) and David Nofre, "Unraveling Algol: US, Europe, and the Creation of a Programming Language," *IEEE Annals of the History of Computing* 32/2 (2010): 58-68. ↩
10. Transl. by the Author. See Heinz Rutishauser, "Massnahmen zur Vereinfachung des Programmierens. (Bericht über die 5-jährige Programmierarbeit mit der Z4 gewonnenen Erfahrungen)," in *Elektronische Rechenmaschinen und Informationsverarbeitung: Nachrichtentechnische Fachberichte* 4, (1956): 28. ↩
11. Donald E. Knuth / Luis T. Pardo, "The Early Development of Programming Languages," in *A History of Computing in the Twentieth Century. A collection of essays with introductory essay and indexes*, ed. N. Metropolis, J. Howlett and Gian-Carlo Rota (New York:

Academic Press, 1980), 206. ↩
12. See for example Goldschlager/ Lister 1982, 179. ↩
13. See W. Nijenhuis, “Listening in to the PASCAL,” *Philips Technical Review* 24/4-5 (1962/63): 164-170. Special thanks to Dr. Albert Gerards, University of Amsterdam for mentioning this valuable source. The audio file is accessible via (link to audiofile). ↩
14. See generally Wolfgang Ernst, “Media Archaeography. Method and Machine versus History and Narrative of Media,” in *Media Archaeology. Approaches, Applications, and Implications*, ed. Erkki Huhtamo and Jussi Parikka (Berkeley, Calif.: University of California Press, 2011), 239-255. ↩
15. Nijenhuis, “Listening in to the PASCAL,” 174. ↩
16. UNIVAC Conference, OH 200. Oral history on 17-18 May 1990. Smithsonian Institution Washington, D.C. (University of Minnesota, Minneapolis: Charles Babbage Institute), 72. ↩
17. An example of an early operation systems is the Compatible Time-Sharing System developed by the MIT in the early 1960s. See Fernando J. Corbató et al, *The Compatible Time-Sharing System. A Programmer’s Guide*, (Cambridge, MA: MIT Press, 1963). ↩
18. See on the notion of implicit and tacit knowledge Michael Polanyi, *The Tacit Dimension*, (Garden City, NY: Doubleday & Company Inc., 1966), 3-25. ↩
19. See for an exception Frederic C. Williams / Tom Kilburn, “The University of Manchester Computing Machine,” *Proceedings of the Review of Electronic Digital Computers. International Workshop on Managing Requirements Knowledge* (1951): 57. ↩
20. Matthew Kirschenbaum, *Mechanism. New Media and the Forensic Imagination*, (Cambridge, MA: MIT Press, 2008), 10ff. ↩
21. See generally Adrian Mackenzie, *Wirelessness. Radical Empiricism in Network Cultures*, (Cambridge, MA: MIT Press, 2010). ↩
22. John W. Tukey, “The Teaching of Concrete Mathematics”, *The American Mathematical Monthly* 65/1 (1958): 2. ↩
23. See generally Gilles Deleuze and Felix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, transl. Brian Massumi, (London/ New York: Continuum, 2004), 552. ↩
24. See generally Janet Abbate, *Inventing the Internet*, (Cambridge, MA: MIT Press, 1999). ↩
25. John Phillips, “Agencement/Assemblage,” *Theory, Culture & Society* 23/2-3 (2006): 108-109. ↩
26. Michel Callon, “What does it mean to say that economics is performative ?,” in *Do economists make markets? On the performativity of economics*, ed. Donald MacKenzie, Fabian Muniesa and Lucia Siu (New Jersey: Princeton Univeristy Press, 2007), 319ff. and as well Iain Hardie and Donald MacKenzie, “Assembling an economic actor: the agencement of a Hedge Fund,” *The Sociological Review* 55/1 Feb. (2007): 58. ↩
27. Bruno Latour, “On recalling ANT,” in *Actor Network Theory and After*, ed. John Law and John Hassard (Oxford: Blackwell, 1999), 15-25. ↩
28. See for example John Johnston’s usage in, *The Allure of Machinic Life. Cybernetics, Artificial Life, and the New AI*, (Cambridge, MA: MIT Press, 2008), 117. ↩
29. See generally the first chapter in Bruce Sterling, *The Hacker Crackdown*, (New York: Bantam, 1992). ↩
30. Peter G. Neumann, “Risks to the public in computers and related systems,” *ACM SIGSOFT Software Engineering Notes* 15/2 April (1991): 11-13. ↩
31. Neumann 1991, 13. ↩
32. Neumann 1991, 11. ↩
33. Ibid. ↩
34. Jussi Parikka and Tony D. Sampson, “On Anomalous Objects and Digital Culture. An Introduction,” in *The Spam Book. On Viruses, Porn, and Other Anomalies from the Dark Side of Digital Culture*, ed. Jussi Parikka and Tony D. Sampson (Cresskill, NJ: Hampton Press, 2009), 11. ↩
35. Gilbert Simondon, *Du mode d’existence des object techniques*. Thèse complémentaire pour le doctorat és lettres présentée à la Faculté des Lettres de l’Université de Paris, (Paris: Aubier (Edition Montaigne), 1958), 76. ↩
36. AlgorhythmicSorting is such a software project created by the author in collaboration with Michael Chinen. It was released between June 2011 and Jan. 2011 and is accessible on sourcefourge.net. (http://sourceforge.net/projects/algorhythmics/files/, last checked April 2012). ↩
37. N. Katherine Hayles, *How We Became Posthuman. Virtual Bodies in Cybernetics, Literature, and Informatics*, (Chicago: University of Chicago Press, 1999), 30. ↩

Series Navigation

<< What is in PageRank? A Historical and Conceptual Investigation of a Recursive Status Index.The Order of Places: Code, Ontology and Visibility in Locative Media >>