

**T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
SANAT VE TASARIM ANA BİLİM DALI
SANAT VE TASARIM PROGRAMI**

YÜKSEK LİSANS TEZİ

**A MULTISCALE COMPOSITION METHOD BASED ON IMAGE
SONIFICATION**

**ALİ SERKAN SEVİLGİN
19740014**

**TEZ DANIŞMANI
PROF. DR. ARDA EDEN**

2023

**T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
SANAT VE TASARIM ANA BİLİM DALI
SANAT VE TASARIM PROGRAMI**

YÜKSEK LİSANS TEZİ

**A MULTISCALE COMPOSITION METHOD BASED ON IMAGE
SONIFICATION**

**ALİ SERKAN SEVİLGİN
19740014
ORCID NO: 0000-0002-3117-4826**

**TEZ DANIŞMANI
PROF. DR. ARDA EDEN**

OCAK 2023

Ali Serkan Sevilgen tarafından hazırlanan “A MULTISCALE COMPOSITION METHOD BASED ON IMAGE SONIFICATION” başlıklı çalışma, **03/07/2023** tarihinde yapılan savunma sınavı sonucunda oybirliği ile başarılı bulunmuş ve jürimiz tarafından Sanat ve Tasarım Ana Bilim Dalı, Sanat ve Tasarım Programında **YÜKSEK LİSANS** tezi olarak kabul edilmiştir.

Danışman

İmza

PROF. DR. ARDA EDEN

.....

Jüri Üyeleri

İmza

DR.ÖĞR.ÜYE. ECE MERVE YÜCEER NISHIDA

DR.ÖĞR.ÜYE. GÖKHAN DENEÇ

ABSTRACT

A MULTISCALE COMPOSITION METHOD BASED ON IMAGE SONIFICATION

This thesis explores the utilization of data sonification for music composition, specifically focusing on the challenges associated with creatively sonifying data in relation to musical structure. It seeks to diverge from the entrenched bottom-up method common in generative music, and instead, builds the foundation for multiscale formal strategies within sonification. The proposed compositional approach draws inspiration from Iannis Xenakis' (1922 - 2001) theories and techniques, particularly his approach in the piece *Achorripsis* (1957). In this piece, Xenakis employs a matrix in which columns symbolize time, and rows represent timbral groups. Despite leveraging algorithmic methods to generate music, Xenakis circumvents the drawbacks of the bottom-up technique, maintaining the music form's control. Notably, this matrix closely mirrors a digital image, essentially an array of pixel values arranged in a rectangular grid, where each cell or pixel cluster contains data ripe for sonification. To conserve the intricate relations inherent in images, machine learning algorithms are deployed for segmentation. The custom-built software, based on Python, a leading platform for data analysis and machine learning, processes the input data, which is subsequently sonified through the Csound audio programming platform. Utilizing digital images in this manner offers several advantages over other data sonification methods: Digital image data's well-defined structure facilitates automation of tedious tasks like cleaning, restructuring, and fixing, enhancing the replicability of the research with various data sources; an average-sized image file supplies abundant data points; it allows easy access to a diverse array of image files; and current research in digital image analysis and processing, inclusive of machine learning, holds the potential to catalyze fresh opportunities for this project. Musical pieces crafted using this technique pave the way for further exploration of the method's adaptability to the research question.

Keywords: Sonification, Algorithmic Music, Musical Form, Iannis Xenakis

ÖZET

GÖRÜNTÜ İLE VERİ SESLEME ÜZERİNE ÇOK ÖLÇEKLİ BİR BESTELEME YÖNTEMİ

Bu yüksek lisans tezi bir müzik besteleme yöntemi olarak veri seslendirme hakkında bir araştırma sunar. Veri seslendirmenin yaratıcı düzeyde kullanımı, özellikle de müziksel biçim ile ilişkisi tartışılmaktadır. Amaç, algoritmik müzik üretimine içkin olan “aşağıdan-yukarı” (“bottom-up”) biçim oluşturma yönteminden kaçınma ve veri seslendirmede çok ölçekli biçim stratejileri için bir alan yaratmaktır. Önerilen besteleme yönteminde, besteci Iannis Xenakis’in (1922 - 2001) teorilerinden ve kullandığı tekniklerden esinlenilmiştir. Xenakis, *Achorripsis* (1957) isimli eserinde sütunları, zamanı, satırları, tını gruplarını simgeleyen bir matriks oluşturmuştur. Algoritmik yordamları kullanmasına rağmen “aşağıdan-yukarı” yaklaşımdan uzak durabilmiş ve besteci olarak biçim üzerinde hakimiyet sağlamıştır. Bu tez çalışmasında Xenakis’in kurduğu bu matriksin, piksel değerlerinden oluşan bir dikdörtgen yapı olan sayısal görüntü ile büyük benzerlikleri tespit edilmiştir. Sayısal görüntüde, veri seslendirmeye tabi tutabileceğimiz piksel ya da piksel gruplarından oluşan hücreler, Xenakis’in matriks modellemesine benzer şekilde uygulamaya hazırdır. Bu tez çalışmasında sayısal görüntüdeki bölümlendirmede yüksek seviye ilişkileri korumak için bir makine öğrenmesi algoritması kullanılmıştır. Yazılım dili olarak veri işleme ve makine öğrenmesi alanlarında yaygın kullanılan Python programlama dili tercih edilmiş ve işlenen verinin sese dönüştürülmesi aşmasında Csound ses programlama dili kullanılmıştır. Önerilen besteleme yönteminde veri kaynağı olarak sayısal görüntü alınmasından kaynaklanan avantajlar değerlendirilmiştir. Sayısal görüntü verisinin standard bir yapıya sahip olması bize temizleme, düzeltme, yeniden yapılandırma gibi el oyalayıcı işlemleri otomatize etme olanağı sağlamıştır. Bu işlem farklı kaynaklarla aynı yöntemin tekrarlanmasını mümkün kılar. Ortalama büyüklükte bir görüntü dosyası yeterince veri noktası sağlar. Aynı zamanda çok sayıda görüntü dosyasına erişim de mümkündür. Bu çalışmada öngördüğümüz üzere, görüntü işleme ve analizi alanlarındaki gelişmeler, yapay zeka yöntemleri de dahil olmak üzere zamanla pek çok yeni imkanlar sunacaktır. Çalışmamızda sunduğumuz yöntemle bestelenen müzikler, yöntemin araştırma sorusuna uygulanabilirliğini de tartışmaya açmamıza imkan sağlayacaktır.

Anahtar Kelimeler: Veri seslendirme, Algoritmik Müzik, Müzikal Form, Iannis Xenakis

PREFACE

I am pleased to present this Master's thesis, which explores data sonification as a music composition method. This research journey was made possible by the invaluable support and guidance of my academic advisor, Arda Eden. I am deeply grateful to my family for their unwavering encouragement, and to Nihan Tahtaşleyen for her understanding and support throughout this endeavor. Their contributions have played a crucial role in shaping this work.

Ali Serkan Sevilgen

July, 2023; İstanbul

CONTENTS

ABSTRACT.....	iii
OZET.....	iv
PREFACE.....	v
CONTENTS.....	vi
LIST OF TABLES.....	vii
LIST OF FIGURES.....	viii
LIST OF PROGRAM CODE.....	ix
1. INTRODUCTION.....	1
1.1. Sonification.....	1
1.1.1. Methods of Sonification.....	3
1.1.2. Examples of Sonification.....	6
1.1.3. Aesthetics.....	8
1.2. Musical Form.....	11
1.2.1. Form in Algorithmic Music.....	15
1.2.2. Xenakis and Form.....	18
1.3. <i>Achorripsis</i>	20
2. METHODOLOGY.....	22
2.1. Formalization of Form.....	22
2.2. Image as a Data Source.....	23
2.3. Xenakis's Matrix to Image Matrix.....	23
2.4. Mapping Strategies.....	28
2.5. Phases of Image Sonification.....	28
2.6. Components.....	29
2.7. Composition Plan.....	31
3. MUSIC PIECES.....	31
3.1. Granular for Recorded Cello.....	31
3.2. Gendy Cloud.....	33
3.3. Mar8.....	40
4. CONCLUSION.....	44
REFERENCES.....	45
A SOURCE CODES.....	52
A.1. Granular for Cello Python Code.....	52
A.2. Mar8 Csound Code.....	55
A.3. Gendy Cloud Csound Code.....	61

LIST OF TABLES

Table 1. Listening Modes In Auditory Display And Musique Concrete	9
Table 2. Data-Driven Music Methods.....	10
Table 3. Mapping Preferences In The Literature	12
Table 4. Form Building With Pattern Generator.....	33



LIST OF FIGURES

Figure 1.1. Sonification Workflow.	4
Figure 1.2. Relation Of Artistic Expression And Data Representation.....	9
Figure 1.3. Music And Sonification	10
Figure 1.4. Distribution Of The Sonification Projects	12
Figure 1.5. Sonic Material Used In The Sonification Projects.	13
Figure 1.6. Supper's Loop Of Construction Evaluation.....	17
Figure 1.7. Three Compositional Strategies	18
Figure 1.8. The Matrix That Xenakis Designed For Achorriopsis.....	21
Figure 2.1. Achorriopsis Matrix Recreated With Color-Coding	23
Figure 2.2. Original Image.....	24
Figure 2.3. Image Pixel Values Are Averaged	25
Figure 2.4. Image With Contour-1.....	26
Figure 2.5. Image With Contour-2.	27
Figure 2.6. A Test Of Segmentation By Slc Algorithm-1.	27
Figure 2.7. A Test Of Segmentation By Slc Algorithm-2.....	28
Figure 3.1. Segmentation With Slc Algorithm At Different Levels	32
Figure 3.2. Image Data/Sound Parameters Mapping	33
Figure 3.3. Visualization Of Extracted Data From The Image For Each Parameter And Selected Segmentation.	34
Figure 3.4. 30 Superpixel Segmentation For Gendy Cloud.....	36
Figure 3.5. 300 Superpixel Segmentation For Gendy Cloud	37
Figure 3.6. 3000 Superpixel Segmentation For Gendy Cloud	37
Figure 3.7. Poisson Lambda Values Assigned To Duration And Rest Values For 8 Controller With 300 Segment Image.....	37
Figure 3.8. Data Extracted For Eight Pattern Controller	38
Figure 3.9. 30 Super Pixel Segmentation For Mar8.....	40
Figure 3.10. 300 Superpixel Segmentation For Mar8	41
Figure 3.11 3000 Superpixel Segmentation For Mar8	41
Figure 3.12 Extracting Data With Several Segmentation Strategies For Mar8	42
Figure 3.13 Visualization Of Data Extracted From The Image For Mar8.....	43

LIST OF PROGRAM CODE

1. A Simple Sine Wave Oscillator With Amp And Freq	5
2. A Simple Csound Code.....	30
3. Instrument Code	35
4. Csound Score Sample.....	36
4. Gendy Cloud Gendyn Based Csound Instrument.....	39



1 INTRODUCTION

This thesis presents research on data sonification as a music composition method. The problems of sonification in a creative manner are discussed, especially its relation to musical form. Sonification is regarded as a subset of algorithmic music and shares similar concerns regarding form. The method of the bottom-up approach is the consequence of algorithmic procedures. That means the form is the result of low-level musical activities which are bound to an algorithm that is employed. Form building which is an essential part of the composition process, is left out. The composer has only indirect and limited control over the form. The form is determined by data or data that the algorithm produces. The question is how we can employ other form strategies like top-down or multi-scale form organizations with sonification. The inspiration for a solution comes from the composer Iannis Xenakis (1922 - 2001). In his book *Formalized Music* (Xenakis, 1992), he describes the "Fundamental Phases of a Musical Work" that can be regarded as a framework for the compositional process. Even though Xenakis is regarded to be the founder of algorithmic music, the process is hardly a bottom-up approach or a top-down approach. He emphasizes going back and forth between micro-composition and macro-composition. His music piece *Achorripsis* is a good example of his approach, and luckily he describes his methodology in *Formalized Music*. This helps us to evaluate his composition method with an example of its suitability for our form, algorithmic music relation problem.

1.1 Sonification

Sonification is a relatively new field that emerged from studies in auditory display. "Auditory Display refers to the use of sound to display data, monitor systems, and enhance user interfaces." (Kramer & Walker, 2005). The International Conference on Auditory Display (ICAD) is the main event that researchers gather annually. It is attempted to define sonification by several researchers. I would like to present a few of them here to bring up their similarities and differences.

"Sonification is defined as the use of nonspeech audio to convey information. More specifically, sonification is the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation." (Kramer et al., 1999) The mentioning of "nonspeech audio" is important because, based on the definition, the communicative aspect of sound is the audio itself not a verbal one.

Hermann defines sonification as "a technique that uses data as input and generates sound signals" under certain conditions.

[A technique] may be called sonification, if and only if

1. The sound reflects objective properties or relations in the input data.
2. The transformation is systematic. This means that there is a precise definition provided of how the data (and optional interactions) cause the sound to change.
3. The sonification is reproducible: given the same data and identical interactions (or triggers), the resulting sound has to be structurally identical.
4. The system can intentionally be used with different data and also be used in repetition with the same data. (Hermann, 2008, p. 2)

Scaletti (2018) suggests that sonification is "a mapping of numerically represented relations in some domain under study to relations in an acoustic domain for the purpose of interpreting, understanding, or communicating relations in the domain under study." (Scaletti, 2018) Barras and Vickers (2011) put it as "Sonification is a rendering of data to sound with the purpose of allowing insight into the data and knowledge generation about the system from which the data is gathered" (Barras & Vickers, 2011)

The main elements of a sonification process start with data and end with audio. And the audio is expected to present the relations of data. The emphasis is common in all definitions that the recipient should understand the relations of original data.

There are strong parallels in relation to sonification to data visualization in terms of the goal of the process. Data visualization helps "communicating complex messages through simplified yet powerful representations of otherwise opaque data." (Ridley & Birchall, 2020) Any professional who has to work with data should admit that any job always starts with the understanding of data, and visualization plays an important role. With the vast and complex data, it becomes an essential part of the job. However, there is a difference between visual and aural perception. The former is almost instantaneous; we see things as a whole, even if we don't get the details right away. The latter is temporal. We need to stay focused for a period of time and use our memory to reconstruct what we hear. When we plot data, we can see the changes in data as trends. The two-dimensional visualization can represent any value against any parameter. It can be GDP per country or inflation per year. In sonification, our

parameter axis (usually the x-axis) should always be time. Visualization can represent the same data in different formats like scatter plots, bars, maps, Venn diagrams, and so on. (Hansen & Johnson, 2005) The established tools in the area of "Exploratory Data Analysis" have either built-in visualization libraries like MATLAB (Martinez, 2004) or R (Wickham & Grolemund, 2017) or community supported third party libraries in Python programming language (Embarak, 2018) Each of them has its own purpose and helps understanding of data in a very short time. The one difference is audio itself is complex data, whereas visual tools are very simple. There is an additional psychoacoustic process that is needed to reveal the meaning of the audio data. Because of this, sonification doesn't replace visualization because it is not exactly equivalent to it.

Even though sonification is not comparable to visualization in terms of efficiency, it has its own place where visualization can be less than useful. Visually impaired people can benefit from sonification in various cases. (Blessenohl et al., 2015; Brock & Kristensson, 2013; Bujacz & Strumillo, 2016; Rector et al., 2013) However, competing solutions other than sonification like haptics are also proposed (Kayukawa et al., 2020; Shrewsbury, 2011) In other situations where it is not possible to give full attention to visual clues like sports training, interactive sonification, a sonification method that audio feedback is created based on real-time data input and/or user involvement (Pauletto & Hunt, 2009; Weinberg & Thatcher, 2006) proves as beneficial for improving motor skills of athletes. (O'Brien et al., 2019; Schaffert et al., 2019; Sigrist et al., 2015; van Rheden et al., 2020). In rowing (Dubus & Bresin, 2015; Effenberg et al., 2011; Schaffert & Mattes, 2015), cycling (Schaffert et al., 2017; Sigrist et al., 2016), golf (Ikeda et al., 2019; Tanaka et al., 2018), swimming (Seibert & Hug, 2013), speed-skating (Stienstra et al., 2011) Not only for athletes but also for rehabilitation of patients regarding motor skills, sonification research is gaining momentum (Brückner et al., 2014; Pietschmann et al., 2019; Scholz et al., 2014)

1.1.1 Methods Of Sonification

There are many approaches for applying data to sound conversion. We will explore the three well-defined and mostly employed methods. The first one is *audification*, where data is converted directly into digital audio waves. Digital audio is a discrete signal. The analog signal is sampled according to predetermined fixed intervals, which are called *the sampling rate*. If there are 44100 sample points per second, then the sampling rate is 44.1 kHz. The values of a wave are between -1.0 and 1.0. To accomplish audification, we should

Figure 1.1: Sonification workflow



normalize data for audio format; thus, the maximum would translate into 1.0, and the minimum would translate into -1.0. Then, we should determine the sampling rate. The sampling rate imposes the audio quality. For comparison, the land-line telephone quality is 8 kHz, whereas CD (Compact Disk) quality is 44.1 kHz. In video production, 48 kHz is the standard for audio. Many consumer audio interfaces support 96 kHz or more. This becomes important depending on the size of the data and the desired duration of the audio. For 5 minutes of audio with a 44.1 kHz sampling rate, there is a need for 13,230,000 data points. Scaletti (Scaletti, 2018) proposes an alternative approach if the data size falls short for audification. Instead of converting data into an audio file at the sampling rate, you would convert it at a very low rate to produce very low-frequency audio and then modulate another sound source with data-generated audio. The sound would respect to the changes in the data. Audification assumes the data is one-dimensional means there is only a single value for each data point. If data has more than one dimension, then the others can be flattened and added to the data to be processed. Audio wave format usually follows a smooth line if the source is not noisy. From the reverse, the data can drop and fall due to the nature of the data source. The weather data, such as heat, could change more smoothly; however, the location of a fly would create more leaping values. If the latter is the case, we would expect the audio would be more noisy. Even the former would result in a noisy sound. Thus for the audience, it would be much harder to construct the relations of data from the audio.

The second approach is *parameter mapping*. In this approach, data values are mapped onto musical parameters. This procedure has the potential to generate diverse types of audio and music. We need a sound-producing idea. It could be an acoustic musical instrument like a cello or a digital instrument implemented in an audio programming language. The most basic one is creating a sine wave oscillator and mapping the values of data onto frequency. Our ears are very sensitive to even the slightest changes in sound frequencies. Since a sine wave oscillator generates a simple sound with no harmonics, it has a distinctive quality that can not be found in nature. When the detecting relations of data is crucial, like interactive sonification systems for health or training, this would be the most suitable option. However, listening to a sine wave is not very pleasing, especially for a long period of time. Another idea would be using an analog

instrument and converting data into basic musical parameters like pitch, duration, and intensity. If we use the pitch as in Western tonal music, we should divide the whole spectrum of data into 12 equal parts. This means we are *quantization* our data and losing the variation. In the frequency domain, the result can be very finely granulated due to the nature of frequency data which is *continuous* however, the *pitch* is *discrete*. The first example I will present is fairly simple. Let's assume our data is a series of numbers such as [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. We would decide to map those numbers onto a western chromatic pitch scale like [C, C#, D, D#, E, F, F#, G, G#, A, A#, B]. Even if it seems straightforward, the problem arises when we examine the relations of source data and musical parameters. Our source data implies that the relation between the first and the second value is two-fold. The second value is 2, and the first one is 1, so it is $2 / 1 = 2$. However, in our result data, the first one is the note C, which is 261.63 Hz (cycle per second), and the second one is C# which is 277.18 Hz. Thus the relation here is $277.18 / 261.63 = 1.0594$ instead of 2 in the source data. The second example shows the problems with quantized data. Let's assume the source data is continuous, spanning from 0.000 to 120.000 with a distribution around 4.358 which means we would have lots of data points like 4.234, 4.876, etc. Let's use the same result data from the first example: [C, C#, D, D#, E, F, F#, G, G#, A, A#, B]. In order to map our source data onto result data we should divide it into 12 equal parts, which means we put all data from 0 to 10 in the first window, 10 to 20 to second and so on which results in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]. Then we have the same problem as in the first example so that the relations in data don't overlap and additionally the distribution and the variance in the source data are not reflected in the result data.

`ares oscil xamp, xcps`

Program code 1: A simple sine wave oscillator with amp and freq.

The third method of sonification is *model-based sonification* envisioned by Thomas Hermann (Hermann, 2011). The main objective is to build a virtual model that has the ability to produce sound and using interactive sonification techniques handing the user to explore the possibilities of the system. This approach is closer to algorithmic composition, where the algorithm produces data based on input values. The inspiration comes from the fact that almost any interaction with the world could produce sound. The projects that are developed by Hermann exploit complex data models like the "Growing Neural Gas Sonification Model," "Data Crystallization Sonification," and "Particle Trajectory Sonification Model." His argument is that with those data models,

parameter mapping is not a viable option. The question is whether the audience (or even the performer who excites the model) could follow the relation between the model and the sound.

Here we should stress a problem that is general to sonification. Even if the main goal is communicating data via sound, using it as a narrative device, or building an artistic experience, it should be noted that the choices in the audio domain (either musical instrument, mapping, or the genre of music) would affect the view or *hearing* of the audience. A sonification with non-pleasing sounds would be less successful in transmitting what data include. The definition of *pleasing sounds* would vary according to each audience group. Music appreciation is very diverse and culture-dependent. When sonification is used for artistic purposes, the procedure converges more to the art of music-making. Thus its communication abilities get blurry.

1.1.2 Examples Of Sonification

In various fields, sonification of scientific data is an ongoing research. Sonification is found useful for processing astronomical data when there is a need to analyze complex and large datasets so that the display techniques is inadequate. As Merced-Diaz et al. propose, "For example, when searching for bow-shock and magnetopause crossings, we expect distinctive signatures to be especially apparent above the background noise if the data are sonified." (Diaz-Merced et al., 2011) Efforts to make Electroencephalographic (EEG) data audible started as early as the 1930s (Väljamäe et al., 2013). Ballora suggests the usefulness of heart rate sonification for medical diagnosis. (Ballora et al., 2004) George et al. use sonification to convert climate data into music (George et al., 2017). Their aim is not to help scientists with data analysis but to use music's influence on human emotions.

Miranda et al. (2009) evaluate different synthesis algorithms for sonification and then decide on the combination of granular and additive synthesis based on three criteria "ability to hear the behavior of the data, simplicity of the synthesizer architecture, and interestingness of the results for use in musical compositions" (Miranda et al., 2009) especially the first reason presents that the synthesis technique requires great attention to replicate the interrelations of data in the audible domain. Jensen (Jensen, 2008) deals with non-linear data structure and proposes an aleatoric technique similar to John Cage's infamous I Ching (Jensen, 2009). This also is a good example of using one composer's method to solve problems in sonification. Delatour approaches sonification as using extra-musical material for composition. He starts with

molecular waveforms and then applies Fourier Transform as if they are sound waves to extract pitch information. (Delatour, 2000)

Sturm (2005) gets oceanic data from buoys and applies an audification technique by mapping the spectrum of buoy data onto the audio domain (Sturm, 2005). Kirke (2015) uses a wave tank which is a body of water that generates waves for interactive sonification. Data obtained in real-time from depth monitors and motion tracking of a floating buoy was subjected to sonification via MAX/MSP software. (Kirke et al., 2015)

Polli bases his installation project on the sonification of weather data. He expresses one of the main motivations in sonification "In my artwork, I have tried to develop strategies for the interpretation of data through sound that has both narrative and emotional content because I believe that an emotional connection with data can increase the human understanding and appreciation of the forces at work behind the data." (Polli, 2005)

The composer John Gibson (2006) participated in an exhibit at the United States Botanic Garden that demonstrates the slowness of plant life. "The exhibit features time-lapse videos of plant growth, some at the microscopic level. By speeding up the depiction of plant growth processes, we humans—distracted as we are by our fast-moving, multitasking world—can develop an appreciation for them." Music composed for the exhibition is also slow-moving (Gibson, 2006)

"Increasing public awareness and engagement" is one of the main motives behind the project by Angeler et al., where they exercise data sonification to increase awareness about the unsustainable use of short water supplies and other social-ecological challenges. (Angeler et al., 2018) They document the transformation in agriculture in Spain and translate data into a musical score for a soprano and bass voice.

A good example of using sonification to reap the power of sound as an emotional bearer is an archaeological photography "In the new interpretative possibilities offered by image sonification, the act of translating archaeological information into sound opens up new ways to elicit ... a kind of engagement that aims to move us emotionally and push a fuller, richer engagement with the past." and "Sonifying, ... might be a way of engaging ... a 'heart centred' archaeology" (Graham & Simons, 2021)

Image sonification is a special case where visual data is represented by an auditory signal. Visual data is converted into digital image format, which consists of pixels in two-dimensional space. The problem is replacing a two-dimensional representation with a one-dimensional time domain. One method is to draw a line

across the image and convert the corresponding pixels into sound. The horizontal axis is usually considered as time, and the pixels in the vertical axis are grouped and used for audio mapping. A *query-based method* requires human intervention to select a portion of data to be sonified. A *tour-based method* is similar to a query-based one, but the data direction is predefined instead of interactive. In *path-based method*, a path is a series of points. Those points can be selected randomly or using computer vision techniques. (Sarkar et al., 2012)

VOSIS is an interactive software that can be used with any kind of image data either static or moving. Users can select a region of grayscale pixel information by drawing rectangular boxes or using a touch-screen. Specifying multiple regions results in chords (multiple pitches that sound together). The data coming from the selected region is in the range of 0 - 255, and it is mapped onto -1.0 - 1.0 for obtaining a typical sound wave. This procedure is an example of *audification* (McGee, 2013). *Sonic Panoramas* is an interactive installation that utilizes large-scale landscape images and selects image data by tracking the movements of the participants in the physical space who observe the projected image. At the sonification stage, "data extracted from the images can be used to generate musical structures either on the macro level, as applied to elements such as rhythm and form, or it can be applied on the micro level through sound synthesis and timbre." (Kabisch et al., 2005).

Musical Vision processes input image to reduce color information and converts RGB data into three notes as MIDI data. The intended output is harmonic polyphonic music. (Polo & Sevillano, 2019)

1.1.3 Aesthetics

Barrass and Vickers discuss the aesthetics of the sonification process, comparing it to composing music. (Barrass & Vickers, 2011) They claim that composers' main objective is the experience rather than the revelation of the compositional materials. The distinction between composition and sonification lies behind the intention of the revelation of the phenomenon that algorithm or data carries. Grond and Hermann state, "Sound becomes sonification when it can claim to possess explanatory powers: when it is neither solely music nor serves as mere illustration" (Grond & Hermann, 2012). In the case of using sonification as a tool for composition undoubtedly brings the choice between conveying a meaning and the abstract nature of the musical structure. The composer should lean to more artistic freedom (see figure 1.2)

Vickers and Hogg (2013) elaborate on the relationship between sonification and music further. (Vickers & Hogg, 2013) They start with the idea of listening

Figure 1.2: Relation of artistic expression and data representation.



Source: Barrass and Vickers, 2011, p. 157

modes proposed in the auditory display field (William W. Gaver, 1989), which is influenced by Schaeffer's listening modes later developed by Michel Chion. (Chion, 2019) in the *musique concrete* field. Everyday listening is similar to casual listening, which prioritizes detecting the source of the heard sound. Musical listening resembles Schaeffer's "Reduced Listening," where the acoustic properties of the sound is subject to inquiry as opposed to everyday or casual listening. Semantic listening serves as the interpretation of meaning in a code or a language. (Chion, 2019) Vickers and Hogg state counterpart of this type of listening is "sonification" in the auditory display.

Table 1: Listening modes in auditory display and *musique concrete*

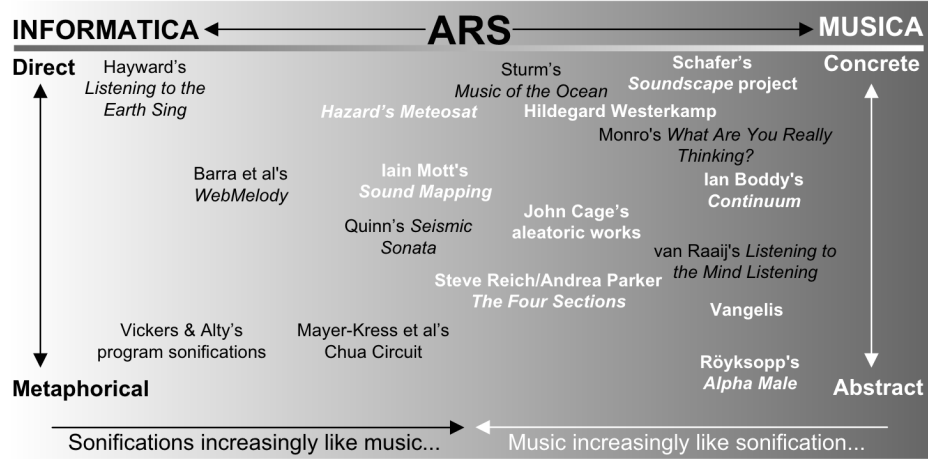
Auditory Display	<i>Musique Concrete</i>
Everyday Listening	Casual Listening
Musical Listening	Reduced Listening
Sonification	Semantic Listening

They later define a two-dimensional is that one dimension being "abstract/concrete" and the other being "sonification/sound art (music)". They propose there is a continuum between the two fields. Tonal, atonal, and serial music have found themselves in the "abstract" category, whereas electroacoustic music concrete is in the "concrete" category. (see figure 1.3)

Ben-Tal and Berger (2004) also suggest the importance of listening for sonification. He seems to find "musical listening" daunting because of the complexity and the emotional associations (probably they actually mean listening to a music piece rather than listening to any sounds in the context of "musical listening") and less daunting when the "listening task is purposeful" like "categorical perception of acoustic signals." He proposes listening is a creative effort to identify trends, anomalies, and patterns. Another important aspect of their research is that their process combines musical composition and sonification "Music and musical ideas served as a starting point for our search for sonification methods, which in turn were reapplied to composition."

Mapping is one of the key problems in computer music and also in sonification. In instrumental music, the score is itself a symbolic representation,

Figure 1.3: Music and Sonification



Source: Vickers and Hogg, 2013, p. 4

and a performer, thanks to intense education and practice, learns how to make the notes on paper audible by his/her instrument. Especially in the "new music," if the composer demands a non-conventional technique of playing, it is described in the legend. The mapping between the score and the sound is intrinsic to the music tradition. In computer music, the composer designs his own instruments using programming. The first problem here is that there is no established symbolic representation, and actually, in many cases, there is no need for it. The code, which is an implemented version of an algorithm, can act as a score generator and/or sound generator. Since there is no intrinsic mapping in this process composer should decide how to map the algorithm (score) into the instruments' performance parameters. This is very similar to the sonification process. An algorithm is a model, and based on its structure, it produces an output according to its inputs. The output of an algorithm is data. In data sonification, we usually refer to "observed data" that is collected or recorded. Only the model is implicit behind the observed data. Our arguments on mapping for computer music or algorithmic music would work for data sonification (2).

Table 2: Data-driven music methods

Sonification	Data -> Sound
Algorithmic Music	Model -> Data -> Sound
Machine Learning	Training Data -> Predicted/Generated Data -> Sound

Mapping can be implemented in two ways: by the use of generative algorithms or by explicitly defined strategies. (Hunt & Wanderley, 2002) The generative approach is more suitable for designing interfaces for performing computer-based instruments (or NIMes - New Interfaces for Musical

Expression) since those types of algorithms, including various machine learning techniques, provide gesture-to-sound translations better. For sonification, the latter can be thought more efficient because it is easier to map two sets of parameters like data values and instrument parameters.

Starting with a set of data values and mapping the synthesis parameters could employ four different strategies: one-to-one, one-to-many, many-to-one, and many-to-many. In data sonification, if the main goal is transparency, then the simplest strategy yields the intended outcome. (Scaletti, 1994) In a scale going to artistic approach, the complex strategies would create more interesting results.

Dubus et al. (2013) present their systematic review of 179 scientific publications comprising 60 sonification projects for their mapping strategies. Parameter mapping is found to be the most popular method for sonification. The auditory dimensions (sound parameters) are grouped into five categories pitch-related, timbral, loudness-related, spatial, and temporal. The pitch is the most used parameter, as one expects. The other two top choices are loudness and duration. (table 3) The pitch, loudness, and duration are the basic parameters to define a musical note. They are also standard in MIDI specification along with *tempo*. Considering the number of projects that uses MIDI for sonic material (see figure 1.5), the mapping choices are strongly related to the tools that researchers employ. The other auditory dimensions they found can be grouped under an umbrella category as *timbre* except spatialization. The spatialization is interesting and hints at electroacoustic music; since there is no spatial information in the electroacoustic domain, it is the composers' role to build acoustic space. Also, the sum of computer music environments like Max/MSP, Pure Data, and SuperCollider used in projects presents electroacoustic music and sonification shares the tools and techniques. "The art and aesthetics" is the second most popular function in their review. (figure 1.4) (Dubus & Bresin, 2013)

1.2 Musical Form

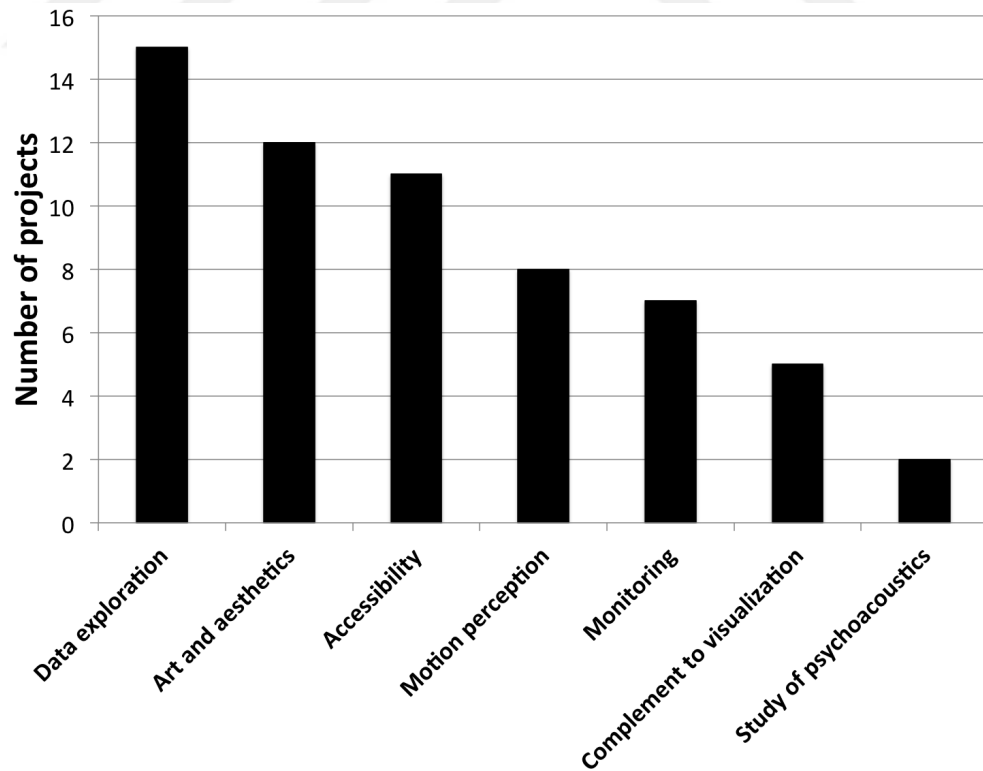
Composition is putting pieces together to build a larger entity. Music composition is no different. The musical materials are brought together to present a higher level of musical expression. I can deduce the composition process into a basic algorithm based on my experience. *composer starts with an initial idea. Then he can repeat the idea as it is, modify it or present a new idea. Then elaborates further.* This resembles the theme generation in the sonata form in the European Art Music tradition. (Caplin, 2013)

It is no surprise that musical form and musical composition are tied

Table 3: Mapping preferences in the literature.

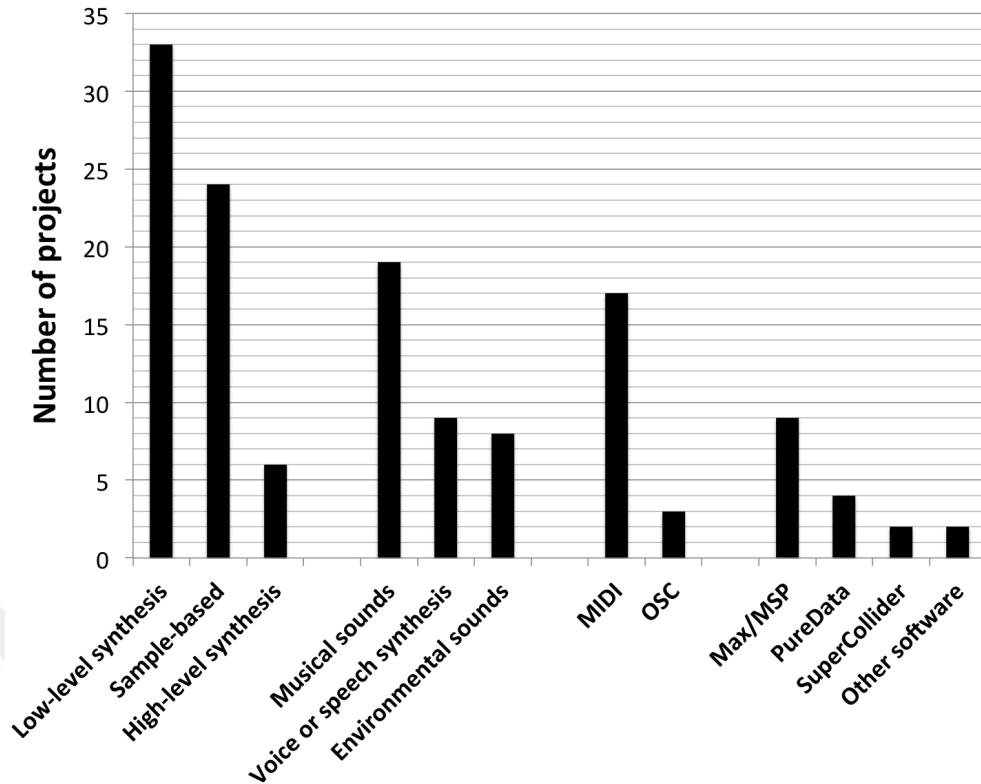
Auditory dimension	Percentage of the total number of mappings
Pitch	23.8
Loudness	15.2
Duration	10.1
Spatialization	9.5
Tempo	5.9
Brightness	5.1
Timbre	3.6
Instrumentation	3.6
Spectral power	2.8
Spectral duration	2.4
Pitch range	2.0
Center frequency of filter	2.0

Figure 1.4: Distribution of the sonification projects according to their primary function.



Source: Dubus and Bresin, 2013, p. 22

Figure 1.5: Sonic material used in the sonification projects.



Source: Dubus and Bresin, 2013, p. 22

together. "Form is only the form of something that has been formed." The well-defined musical forms like rondo, sonata, etc., of the common practice era are strictly bounded to the functional tonal harmony, which is a set of rules to gather sounds successively or simultaneously. With the dissolution of tonal idioms starting from the early 20th century, the existing forms are rendered obsolete or at least anachronic. However, the advances in music which lead to different genres in "new music" or *avant-garde* music are rarely accompanied by new dimensions in formal design (Adorno & Gillespie, 1993).

We should clarify that the *form* we are discussing is the process of composition, not a course of study to analyze a music piece. The latter is, of course, an important practice to understand how a piece is constructed and have the ability to tell more about the music since our memory is limited to consuming lengthy and complex music pieces. Unlike in other sound-related art forms like walkthrough sound installations or site-specific environments, musical compositions have a predefined duration, and the audience is supposed to be experiencing the whole sonic phenomena from start to finish. Formal elements of the music are essential for the audience's perception. (Salter et al., 2008)

Arnold Schönberg, a composer and theorist who is acknowledged as the founder of the serial music that defies tonal harmony, almost only discusses

musical forms in his book, *The Fundamentals of Musical Composition* (1999). He discusses the concept of form very briefly and states that form is the organization of music. The reason that we need form is that form provides *logic* which is the relationship between the presentation of ideas, their importance, and their function, and *coherence* which is useful for avoiding mental limitations for comprehension of a prolonged duration of a music piece. (Schönberg, 1999).

He [composer] conceives an entire composition as a spontaneous vision. Then he proceeds, like Michelangelo who chiselled his *Moses* out of the marble without sketches, complete in every detail, thus directly *forming* the material. (Schönberg, 1999)

We will focus on where Schönberg reveals that the act of composing is forming the material. Thus the form is considered to be the "chisel," the main tool for the composer.

The composer Edgard Varese defined his music as "organized sound" (Varèse & Wen-chung, 1966, p. 18), which brings us back to the Schönberg, who stated form is an organization. Thus the relationship between form and music composition revolves around the problem of organization, which can also make the distinction between them blurry. According to Varese, form is a result of a process, and he explains the process as follows "There is an idea, the basis of an internal structure, expanded and split into different shapes or groups of sound constantly changing in shape, direction, and speed, attracted and repulsed by various forces. The form of the work is the consequence of this interaction" (Varèse & Wen-chung, 1966, p. 11). In relation to the content and form, he claims that they are inseparable, and omitting one of them makes the other nonexistent. (Varèse & Wen-chung, 1966, p. 11)

Young (2016) defines form as the states and transformations of the content. "form embraces characteristics of global shaping and its relationship to short-term clusters of events – gestures, phrases and the dynamic that exists between them." (Young, 2016, p. 59).

According to Miranda (2001), form is a cyclic succession of events. He suggests three levels of abstraction: the microscopic level, the note level, and the building-block level.

the microscopic level deals directly with physical sound attributes, such as frequency, amplitude, spectrum, and so on, which are mainly used for synthesis and sound processing. At the note level, we bundle certain sound attributes together and think of them as a note. Aggregates such as phrases, motives, and structures constitute the building blocks for composition at a higher level

of abstraction. (Miranda, 2001, p. 4)

1.2.1 Form In Algorithmic Music

Algorithm is a "predefined series of steps, and a set of rules or constraints" (Copley & Gartland-Jones, 2005, p. 227). Any musical composition attempt includes a sort of rules or guides to follow. From the broader perspective, we can suggest that *compositional process*, *algorithms*, and *form (as a constructive tool)* are terms that can be used interchangeably depending on the musical context. Roads (Roads, 2015) defines form-building strategies as compositional plans.

Form in the common practice era has the power to determine the overall structure and relations at note-level and motive, phrase-like formal entities level. As an example, a sonata form can be considered an algorithm since it imposes the constraints and affordances that a musical piece can utilize. The sonata form can be seen as an algorithm (Copley & Gartland-Jones, 2005) as well as a compositional process and form (a constructive tool and also the resultant form) at the same time.

The problem of form was neglected in avant-garde music in general and in algorithmic music specifically. The twelve-tone composition technique was developed to replace the functional harmony to define new means of constructing relations between notes. However, the new formal formula or a compositional process, a *modus operandi* didn't flourish, and this task is assigned to composers.

The twelve-tone technique, or *serialism* in general, adopts the rule that a pitch does not repeat itself until all others are heard. Since the possible number of tone rows is limited due to serial row construction techniques, the music pieces can include limited tones. Anton Webern stayed true to the dictated form by the composition technique and composed very short pieces.

Xenakis (1992) criticizes serialism and defines its process as *linear polyphony* before introducing his composition method based on probabilistic theory.

Linear polyphony destroys itself by its very complexity; what one hears is in reality nothing but a mass of notes in various registers. The enormous complexity prevents the audience from following the intertwining of the lines and has as its macroscopic effect an irrational and fortuitous dispersion of sounds over the whole extent of the sonic spectrum. (p. 8)

Hedelin (2008) emphasizes the weakness of algorithmic music in building musical form. He defines the *form* as the final musical structure and *construction* as the algorithm or process of creating music. In algorithmic music, the construction, the algorithm is the starting point. Conclusively,

making the form is insufficient to present the construction, which implies the form is the unintentional result of the algorithmic process. He criticizes that the algorithms used are mostly extra-musical algorithms from natural sciences, mathematics, and computer sciences resulting in poor musical translation. We can conclude that sonification also suffers the very same problem since the methodology and tools of sonification and algorithmic music are similar.

Along with a deficient correlation between construction and form, this points to two inherent weaknesses in algorithmic composition: (1) Weaknesses of the algorithm from a musical perspective – if the construction process does not take into account the phenomena that appear in a musical context, the choice of algorithm is in practice arbitrary. (2) Weaknesses in the mapping of a musical structure if the construction process only takes certain aspects of the music into account, the artistic demands on the algorithm are undermined. (Hedelin, 2008, p. 250)

He proposes a method and develops a software framework, *Kimon*, written in Lisp, to discern the construction and form by choosing the construction and formal characteristics of the music as a starting point. He exemplifies this process with his music piece *Djungle* where he constructs nested structures from the smallest one as sound to the complete piece as follows.

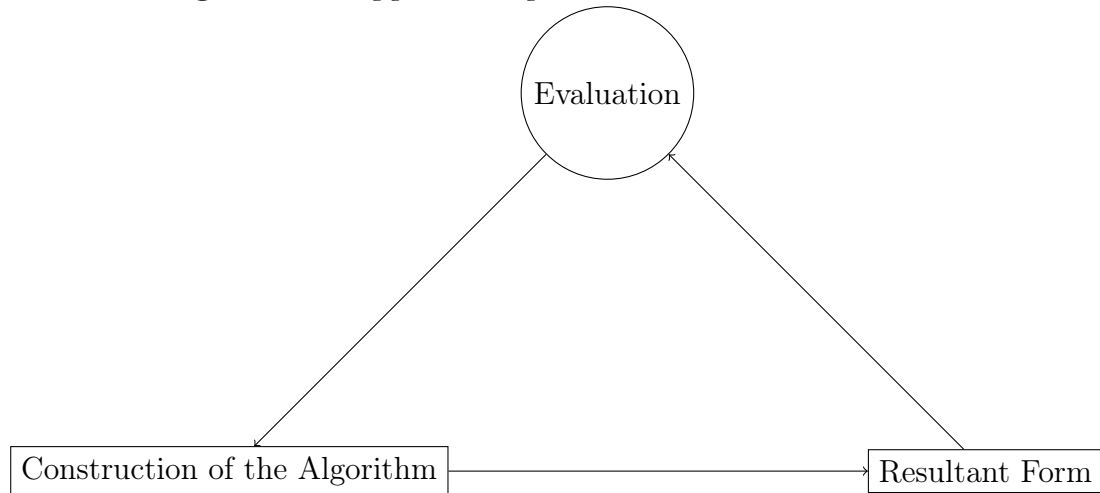
- Sound, the smallest element of the piece.
- Phoneme, comprised of sounds.
- Motive, comprised of phonemes or sounds.
- Phrase, comprised of motives, may contain sounds as a result of interaction.
- Segment, comprised of phrases, may contain motives and sounds as a result of interaction.
- *Djungle*, comprised of segments (Hedelin, 2008, p. 256)

Supper (2001) makes a distinction between the *construction*, which is a compositional idea, and its realization and *resultant form*, which is the auditory perception. He uses the construction as a process (algorithms in the context of algorithmic music) and the resultant form as a finished product. Until the resultant form is as expected musically, the process goes back to construction (fig. 1.6).

He defines three categories for the construction of algorithms.

1. Modeling traditional, non-algorithmic compositional procedures

Figure 1.6: Supper's Loop of Construction Evaluation



Source: Created based on Supper, M. 2001.

2. Modeling new, original compositional procedures, different from those known before
3. Selecting algorithms from extra-musical disciplines (Supper, 2001, p. 48)

There are two main types of form-building strategies: Top-down and bottom-up (Collins, 2009; cite: Miranda, 2001; Narmour, 1991; Roads, 2015)

Miranda states that "algorithmic composition software tends to support the bottom-up approach whereas computer-aided composition software tends to support the top-down approach ... composers tend to combine both approaches" (Miranda, 2001, p. 10) Roads (2015) states that generative music usually employs bottom-up strategies.

Nierhaus suggests that algorithmic procedures can be used for micro and macro composition. "Procedures of algorithmic composition may be used for the treatment of single aspects of a musical task, or for determining the overall structure of a musical piece." (Nierhaus, 2009, p. 259)

Composers usually employ both approaches simultaneously. Roads define this third road as *multiscale* compositional plan.

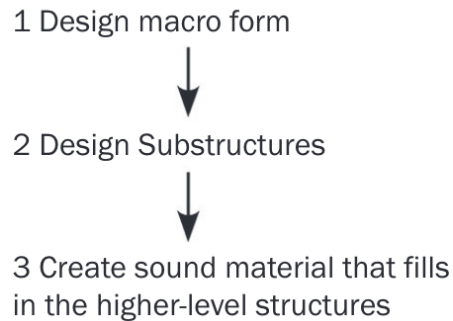
Returning to generative music programs, the following list presents a few possible contenders for novel structural ramifications investigated in systematic composition and mathematical music:

1. Self-similar structures; recursive definitions, fractal music;
2. Stochastic music, information theoretic constructions;
3. Emergent form (as a by-product of complex lower level activity);

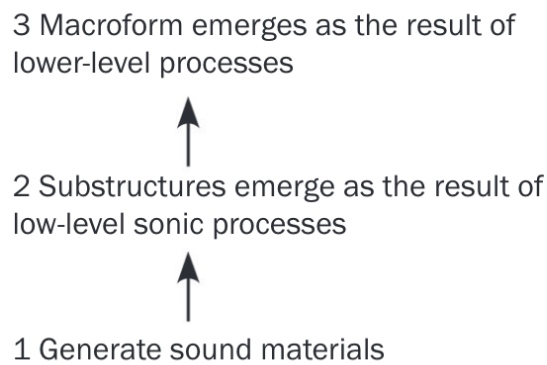
4. Artificial musical grammars. (Collins, 2009)

Figure 1.7: Three compositional strategies

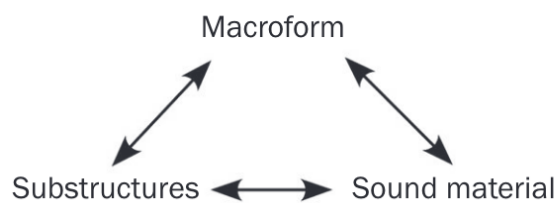
(a) Top-down strategy



(b) Bottom-up strategy



(c) Multiscale planning



Source: Roads, 2015, p. 291

1.2.2 Xenakis And Form

Xenakis expresses his composition process as *FUNDAMENTAL PHASES OF A MUSICAL WORK* as follows

1. *Initial conceptions* (intuitions, provisional or definitive data)
2. *Definition of the sonic entities* and of their symbolism communicable with

the limits of possible means (sounds of musical instruments, electronic sounds, noises, sets of ordered sonic elements, granular or continuous formations, etc.)

3. *Definition of the transformations* which these sonic entities must undergo in the course of the composition (macrocomposition: general choice of logical framework, i.e., of the elementary algebraic operations and the setting up of relations between entities, sets, and their symbols as defined in 2.); and the arrangement of these operations in lexicographic time with the aid of succession and simultaneity)
 - (a) *Microcomposition* (choice and detailed fixing of the functional or stochastic relations of the elements of 2.), i.e., algebra outside-time, and algebra in-time
 - (b) *Sequential programming* of 3. and 4. (the schema and pattern of the work in its entirety)
 - (c) *Implementation of calculations*, verifications, feedbacks, and definitive modifications of the sequential program;
 - (d) *Final symbolic result* of the programming (setting out the music on paper in traditional notation, numerical expressions, graphs, or other means of solfeggio)
 - (e) *Sonic realization* of the program (direct orchestral performance, manipulations of the type of electromagnetic music, computerized construction of the sonic entities and their transformations). (Xenakis, 1992, p. 22)

Even though the macrocomposition phase is established before microcomposition he supposes to have a cyclic event between the two until the work is done. He also comments on the "Phases" and says "The order of this list is not really rigid. Permutations are possible in the course of the working out of a composition." It is apparent that the form is not dependent of the low level relations (bottom-up). However, it is not determined solely by the macrocomposition (top-down). There is not a container that needs to be filled by sonic materials.

In composing *Pithoprakta* (1955–56) the macroform lead the process. (Xenakis & Varga, 1996, p. 75) *Achorripsis* can be seen more like a top-down approach. There is a matrix of events that value of each cell is determined by stochastic processes. At each level down to the sound new process is applied.

In *Achorripsis* I applied the macroscopic approach, from the viewpoint not of

the senses but of the internal structure. It was my aim to create a homogeneous construction based on probabilities which would be interesting for the listener. I realized that probabilities used wrongly could be boring, but if applied properly they could be interesting. (Xenakis & Varga, 1996, p. 79)

In his later works based on *cellular automata* he leans towards bottom-up approach. Cellular automata is similar to Achorripsis Matrix with a grid system comprises vertical and horizontal lines to create cells. He proposes that "It's for the composer (whether working with pictures or with sounds) to fill them." (Xenakis & Varga, 1996, p. 200) It is interesting he sees the parallel between pictures and sounds to fill the cells.

1.3 *Achorripsis*

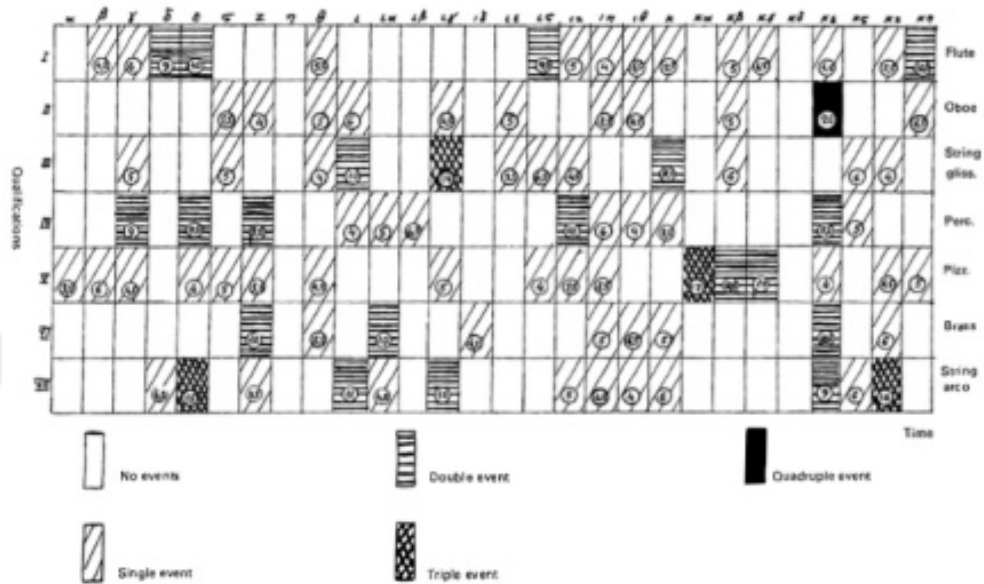
Achorripsis is a piece composed in 1957 and can be seen as the realization of the *Phases of Composition* idea. The piece was composed for 21 instruments which were grouped in seven timbre categories: Flute (piccolo, clarinet, bass clarinet), oboe (oboe, bassoon, contrabassoon), brass (two trumpets, trombone), percussion (xylophone, woodblock, bass drum), pizzicato, string glissandi, string arco (3 violins, 3 violoncellos, 3 contrabasses). He constructed a matrix with 7 columns that each one is dedicated to one of the timbre groups. On the horizontal axis, he used 28 rows as equal time units. The music is planned to endure 7 minutes then each time unit is 15 seconds. Each cell of the matrix was filled with event definitions, like no event, single event, double event, triple event, and quadruple event. Events determine the density of a timbre group at a particular time. (Arsenault, 2002; Childs, 2002; Squibbs et al., n.d.; Xenakis, 1992)

The total number of event types was determined by using the Poisson formula. The Poisson formula is a mathematical equation that calculates the probability of an event based on the Poisson distribution. Xenakis used different probability distributions in his compositions like Poisson, Cauchy, Gaussian, and Uniform distributions. "Poisson distribution is used to model processes where the distribution of the number of incidents occurring in any interval depends only on the length of that interval" (Attenborough, 2003). The lambda, mean value of the distribution was arbitrarily chosen as 0.6 to produce at least one quadruple event, 4 triple events, 19 double events, 65 single events, and 107 zero events. in the 196 cell matrix. The events were spread into cells again using the Poisson formula.

We can follow phases in the compositional realization of *Achorripsis*. Xenakis started with initial ideas (timbres, density of clouds) then created a

macro-compositional model with the matrix. From top to down he meticulously calculated every event until he reached the symbolic output of the music which is notation. The formal structure is not emerged due to the musical mapping of stochastic calculation on to basic vector of a note as pitch, duration, and amplitude.

Figure 1.8: The matrix that Xenakis designed for *Achorripsis*



Source: Xenakis, 1992, p. 23

2 METHODOLOGY

The main motivation of this research is exploring the possibility of multiscale formal organization in data sonification. The "Fundamental Phases" by Xenakis shows that it is possible in algorithmic music. The composer should define micro and macro compositional elements of the music then find or create data that are suitable to map on those parameters. In sonification the data should also be arranged as micro and macro.

Having a proper mechanism to translate the formal structure of data into a music. Music structures can be very complex and in computer music the number of the possible parameters can be high. We need process data points to get data that reflects the higher-level formal structures. We would be able to generate different levels of structures to use in music. Any musical parameter can be mapped onto any level of data structure.

Macrocomposition: High-level data -> High-level musical parameters

Intermediate levels

Microcomposition: Low-level data -> Low-level musical parameters

2.1 Formalization Of Form

We can define the relationship between the composition, form and material as follow

$$C = f(m)$$

where C is composition, f is a function of form and m is the material or content. In this regard we should assume that our function includes the states and also transition logic. Practically, we would have *composition of functions* such as

$$C = f(g(h(m)))$$

where f , g and h are nested functions with same or different methods to process their inputs. Even though it seems like deterministic any stochastic procedure can be implemented in the functions. This type of expression, *formalization of form*, that we propose can be used to define a musical event to desired granularity and extension with its material and form. time can be an argument to the function *form* such as $f(m, t)$ or a compulsory variable within the function

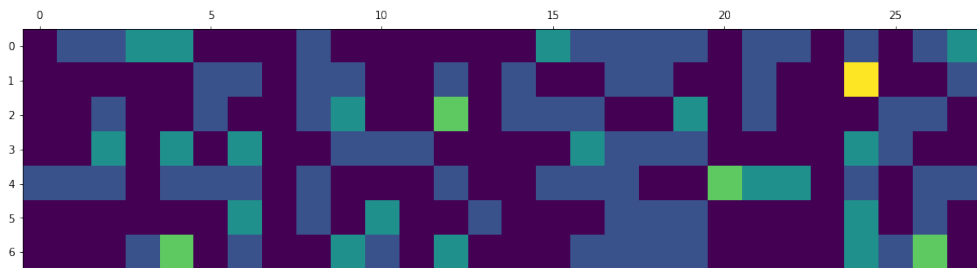
2.2 Image As A Data Source

In order to build a composition model that is based on sonification, we need relatively large data structures so that we can map as many sound parameters, and create mid to large-scale works. A digital image file is made of pixels that are defined as RGB (Red, Green, Blue) values. A 24megapixel camera captures 24 million pixels with each one holding three values for Red, Green, and Blue which yields 72 million data points. For reference, a simple audification (direct converting data to audio) of that data would produce 27 minutes of audio at a 44.1 kHz sampling rate. The resources are abundant and it is easy to access or create digital images that can help to replicate the experiments with different data files. Since image data is already a visualized form of data, browsing for it and further exploration is easy and intuitive. The structure of the digital images is well established. JPEG specification which is a widely used compressed image format was first published in 1992 (Pennebaker & Mitchell, 1992). Software packages and libraries are developed to parse, convert and manipulate different image formats. This capacitates the automation of laborious data preprocessing for successive experiments. However, the sonification of an image is tricky due to its formal structure and perception that differs wildly from music.

2.3 Xenakis's Matrix To Image Matrix

The pixels are arranged as rows and columns in a typical image file. When we parse an image file in a programming environment we create a matrix structure with rows and columns. Reversely we can generate a matrix and then visualize it. If we wish to create an image with 7 rows height and 28 columns width and assign event values like zero, one, two, three, and four to each cell and colorcode them we end up in a two-dimensional space that resembles Xenakis's matrix for *Achorripsis*.

Figure 2.1: *Achorripsis* Matrix recreated with color-coding



The visual similarity between an image matrix and Xenakis's matrix sparks the possibility of transferring strategies that Xenakis employed to image sonification. We can set the timbral qualities on the vertical axis and use the

horizontal axis for durations. Since an average image file allows plenty of data points we can map many of the parameters of instruments. To begin with the toplevel organization of the work, we need to reduce the amount of data. Dividing the time axis equally would result in a pixelated version of the picture which does not present the shapes and patterns we observe in the original picture especially when the pixel size is big. It would also lead to changing the values at the same intervals and synchronously. We need to keep the structural quality of the image while dividing it into sections at different detail levels. The desired process can be illustrated as follows: If we look at a picture from a distance we experience large sections of colors and shapes. When we get closer and closer to the painting new details are revealed at each step until we can see the brushstrokes of the painter, the cracks in the paint layers, subtle color gradients, etc.

The first attempt to get the macro scale information out of an image is to group pixels according to their locations in the image. The arbitrary numbers, the number of parameters for Y-axis and the time intervals for X-axis would be used to generate a matrix very similar to *Achorripsis*. This helped us to diminish the data size into a useful number. The close-up picture of a rusty door from an old building into 100 cells as an 10 to 10 matrix and assign the average value of pixels to a cell. The result (figure 2.3) looks like a matrix similar to *Achorripsis* however we lose most of the formal information of the original picture. From the second image we can not identify the shapes and textures. The visual resemblance between the picture (figure 2.2) and data we obtain is weak. This method works for creating a matrix but fails for translating formal structures of image into data.

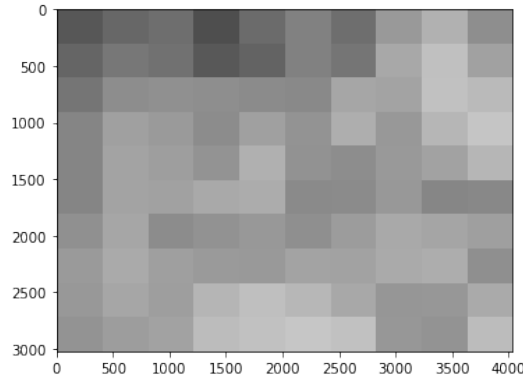
Figure 2.2: Original Image



Source: Image captured by the author

In the second attempt we try to get the contours of the image. This method successfully presents the overall look of an image. However, there is no way to get multi-level of data. We can only get more detailed contour line which is

Figure 2.3: Image Pixel values are averaged



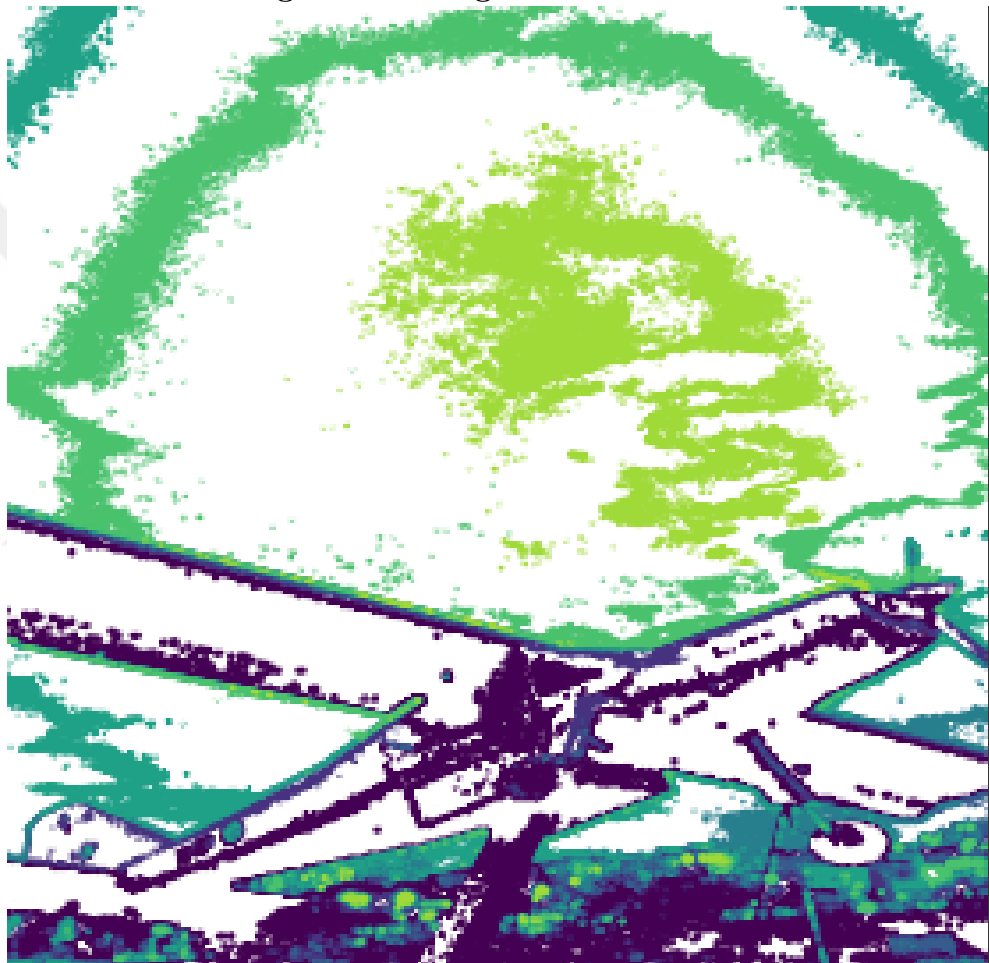
not helpful. We also lose most of the information with discarding non-contour sections.

The third attempt involved the use of research in computer vision. "The field of computer vision is devoted to discovering algorithms, data representations, and computer architectures that embody the principles underlying visual capabilities." (Aloimonos & Rosenfeld, 1991) Even though it has a relatively long history in Artificial Intelligence, it is proved that to replicate human eye, neural system and brain is a complex task (Waldrop, 1984)

Supapixel algorithms can group pixels according to similarities in color, brightness, and other low-level properties at a very high success rate. (Ren & Malik, 2003) Supapixels are powerful to retain the overall look of an image while diminishing data. The size of the superpixels and the number of superpixels that can be found in an image is inversely proportional. If we divide an image into a higher number of sections we will lose more details but we obtain a more general view that we can use for parameters we demand to change less often. We opted to use SLIC (simple linear iterative clustering) superpixel algorithm that utilizes a k-means clustering, a machine learning algorithm. SLIC is simple to use, fast, and successful at detecting boundaries. (Achanta et al., 2012) Speed of the algorithm is important for testing different segmentation strategies. Depending on the arbitrary number we defined for the number of segments, the SLIC algorithm tries its best to identify similar areas. If the chosen number of segments is low then larger areas are selected by the algorithm. Increasing the number produces smaller segments, hence a more detailed picture. Then the next step is averaging the values in each segment. When we draw a line horizontally through the picture we get changing values that we can use for any parameter in our musical design. Thanks to the vast amount of data an image file can provide we can draw as many lines.

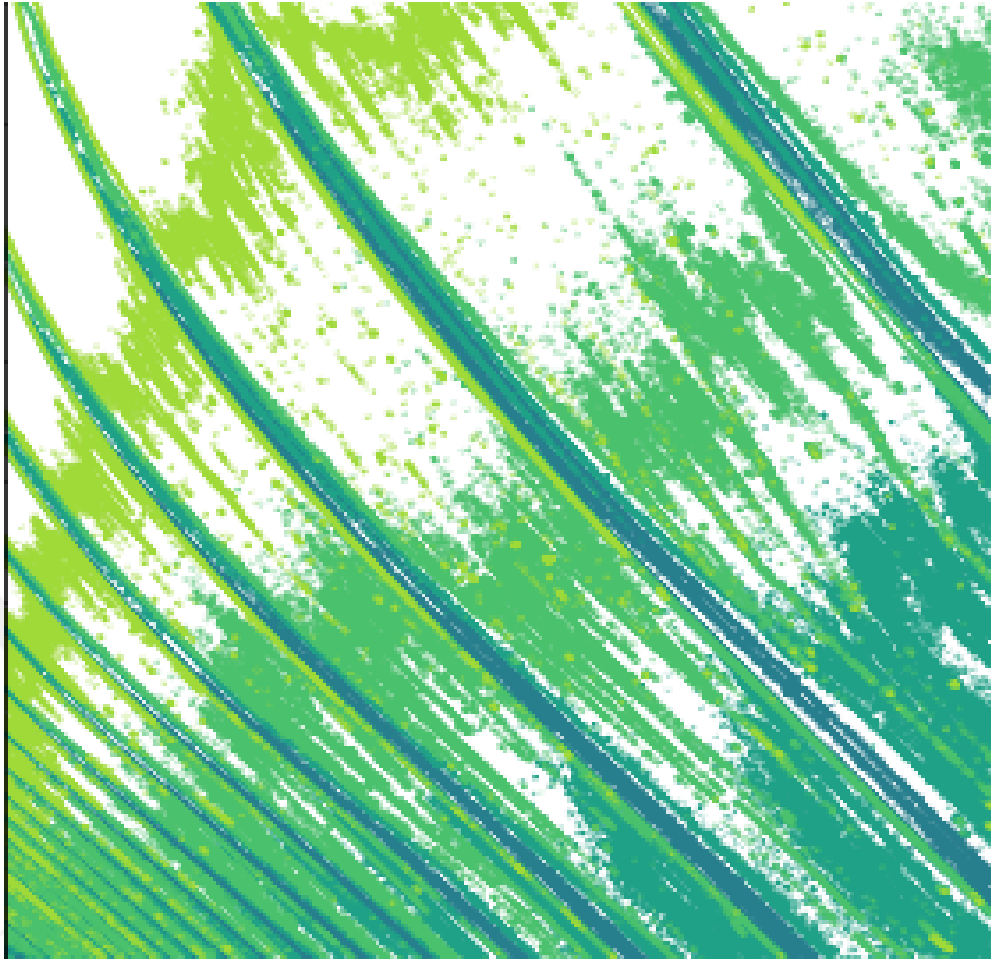
To test our method we've used a picture with 6000 px width and 4000 px

Figure 2.4: Image with contour - 1



Source: Original image captured by the author

Figure 2.5: Image with contour - 2



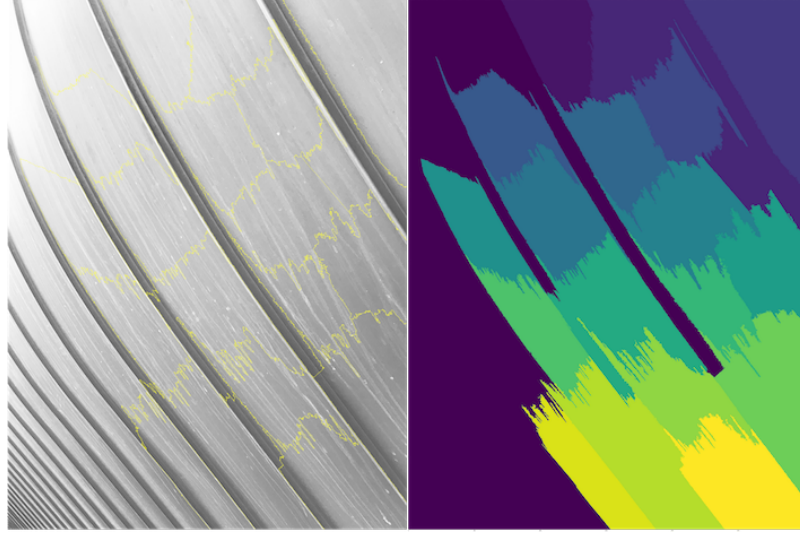
Source: Original image captured by the author

Figure 2.6: A test of segmentation by SLIC algorithm - 1



Source: Original image captured by the author

Figure 2.7: A test of segmentation by SLIC algorithm - 2



Source: Original image captured by the author

height that one of the authors took with the Sony A6000 digital camera. The programming environment was Python which is popular amongst the data science and machine learning community. Using the scikit-image library we were able to implement the SLIC segmentation (van der Walt et al., 2014). We decided to use 30, 300, and 3000 segment numbers and the non-segmented version after experimenting with several other options. 30 segments provided the largest sections as expected. The segment areas got smaller as the segment numbers rose. In image segmentation, the extracted data by drawing horizontal lines on the picture was plotted to visualize the transformation for a better understanding of the result of choices with location and segment numbers.

2.4 Mapping Strategies

Form building is closely related to compositional process. In electroacoustic music when creating an instrument it is up to the composer which properties of the instrument should be static or can be varied over time. In traditional acoustic music the variables are usually dynamics, pitch and duration. It is very unusual to make position of the violin's bridge. However, in electroacoustic music since the composer is also an instrument designer any property can be subject to change.

2.5 Phases Of Image Sonification

Following Xenakis's steps in *Fundamental Phases in Musical Work* closely it is viable to suggest the following procedure for image sonification based composition.

1. Initial conceptions (including finding an image)

2. Definition of the sonic entities (sound materials, synthesis techniques, etc.)
3. Definition of the transformations (Segmentation of the image, mapping strategies)
4. Microcomposition (Csound orchestra programming)
5. Sequential programming of 3. and 4.
6. Implementation of calculations, verifications, etc. (At every step through programming)
7. Final symbolic result of the programming (Csound score)
8. Sonic realization (Rendering Csound score to produce audio file)

2.6 Components

Python is a multi-purpose programming language that is used in data science and machine learning widely. *scikit-image*, the specific library for image processing with machine learning is used for segmentation.

Csound is a powerful programming environment for sound synthesis and signal processing that supports many algorithms (Boulanger, 2000). It has an active developer and user community, and a codebase that was developed throughout the decades. The origin of Csound is **Music N** series software (Music 1 to Music 5) that was developed by Max Matthews in 1957 which was the first digital synthesis software (Manning, 2013). In 1984, Barry Vercoe rewrote the source code in C programming language and the software was named as Csound. Thanks to the long life of Csound almost any new paradigm in digital synthesis was implemented. One of the main strengths of the Csound is that it is relatively easy to render an old code with a new executable. That makes the music can be rerendered in newer machines and devices in longer period of time. Csound can be used as a sole programming environment for sound synthesis and composition or it can be included as an external library into other general purpose programming languages like Python, C, C++, Haskell or Javascript. That provides wide options for a composer/programmer who can utilize advantages of other languages in a project. The Csound executable is supported by many devices like IOS and Android phones, Raspberry PI ¹ or Bela ² micro-computers that extends the usage of Csound beyond regular computers.

A Csound code usually consists of three sections:

¹<https://www.raspberrypi.org>

²<https://bela.io>

1. Header options Assigns runtime variables like audio/midi device, etc.
2. Orchestra That includes Csound instruments which is definitions to produce audio.
3. Score The timeline that determines how the sound will behave over time.

The instruments are defined within *instr .. endin* block. The score defines performance parameters (p-values) which the first three are reserved for instrument number, start time, and duration. Then any number of additional parameters can be assigned as *p4, p5, p6,...* (See figure 2) The structure of score section which is lines of text offers the possibility of generating score with another programming environment. The generated score file than easily can be included in Csound file.

```
<CsoundSynthesizer>
<CsOptions>
-odac
</CsOptions>
<CsInstruments>
instr 1
  kfreq = p4
  aSin poscil 0dbfs/4, kfreq
  out aSin
endin
</CsInstruments>
<CsScore>
i 1 0 5 440
i 1 5 5 880
</CsScore>
</CsoundSynthesizer>
```

Program code 2: A simple Csound code

2.7 Composition Plan

1. Start with an image
2. Convert it into a matrix (with multi-level resolutions to enable jumping between micro and macro decisions)
3. Y-axis for timbral definitions
4. X-axis for timescales

3 MUSIC PIECES

The following music pieces are composed to exemplify the use of the steps described in the methodology. In each piece, data is extracted from an image with Python programming language. Using visualization of data, the number of segments and the position of the line crossed across the image are determined. The synthesis is done in Csound audio programming language. Musical parameters and data points are mapped. This process is not linear; the composer works back and forth until the desired musical output is achieved.

3.1 Granular For Recorded Cello

The piece uses only a single recorded cello sound as an initial material. the sound generation algorithm should have been relatively simple yet versatile. We picked granular synthesis for the project since it can generate variable textures with controlling few parameters. Granular synthesis is a method to get the slice of audio preferably less than 100 ms and then combining those sound portions in large amounts to create cloud-like textures (Roads, 2001). The sound realization of the piece was accomplished in Csound, the audio programming environment. Csound can work in the orchestra (the instrument definitions) and score (note-events) paradigm (Boulanger, 2000) so that we were able to generate a score file in Python as a list of values that describes the changes in instrument parameters. Csound renders the instrument and score definitions as an audio file which can be a short process. This workflow enables us to experiment with several options while keeping the results as Csound score file and audio file for further analysis. The software instrument is based on Csound's grain opcode that we chose to map the grain duration, density, and pitch parameters 1 since they are the most definitive ones on the sonic quality and the texture. The variability on the sound was achieved using separate grain functions for left and right channels while keeping grain, duration, density, and pitch the same but assigning random values for amplitude and pitch offset parameters. The only parameter for the blandly added reverb is the amount of the effect to the dry signal. The instrument accepted the

567 milliseconds of a cello sound as input which was demonstrating a forced bow gesture. Keeping the duration longer resulted in more like a sampler instrument sound against the granular sound which created enticing contrast throughout the piece. The piece was designed to endure 3 minutes for 6000 data points that allow 30 milliseconds for each data point. At each step, values are calculated and written to a text file for rendering in Csound later. The minimum and maximum values for mapping were based on the explicatory needs of the demo piece.

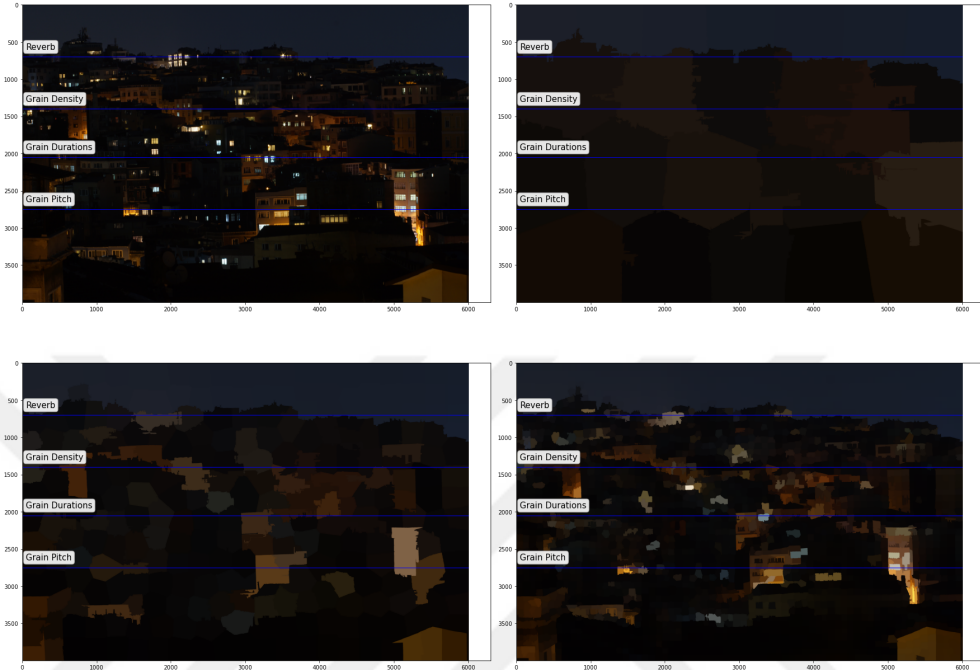
Figure 3.1: Segmentation with SLIC algorithm at different levels



We mapped data onto musical parameters at different time scales with image segmentation. (fig.3 and fig.4) In the course of the piece, parameter values remained the same for the allocated time which resulted in significant formal structures. The generated audio demonstrated that with image sonification it is viable to create a multi-scale form that overcomes the bottom-up approach inherent to sonification and algorithmic music in general. However, we should note that we had to make some design choices to make this experiment easy to process and evaluate the results: The duration of the piece

was rather short; A single instrument with a limited number of parameters was used; Many aspects of electroacoustic music had to be dismissed. The proposed method will be more convincing in large-scale compositions with numerous instruments and parameters.

Figure 3.2: Image data/sound parameters mapping



Source: Original image captured by the author

3.2 Gendy Cloud

Gendy Cloud was originally composed as a networked music piece for real-time, telematic performance. The sound generation is based on dynamic stochastic sound synthesis algorithm, *GENDYN* which was developed by the composer Iannis Xenakis (Hoffmann, 2000). The *GENDYN* algorithm is ported to SuperCollider and Csound programming ³ environments as *Gendy* opcode. Gendy algorithm is capable of producing wide variety of sounds due to its stochastic nature. The ultimate goal of the piece is to generate a sound mass ideally in a multichannel diffusion environment. The result would be a cloud of sounds that the audience would experience different textures at the same time.

The music is rendered by a sound generator instrument (figure 5), a pattern generator, reverb effect instrument and helper functions to store, fetch, and update performance values in an array. There are four main meta-performance parameters to produce music with different formal qualities. The meta performance parameters supply an input for stochastic processes.

³<https://csound.com/docs/manual/gendy.html>

Figure 3.3: Visualization of extracted data from the image for each parameter and selected segmentation

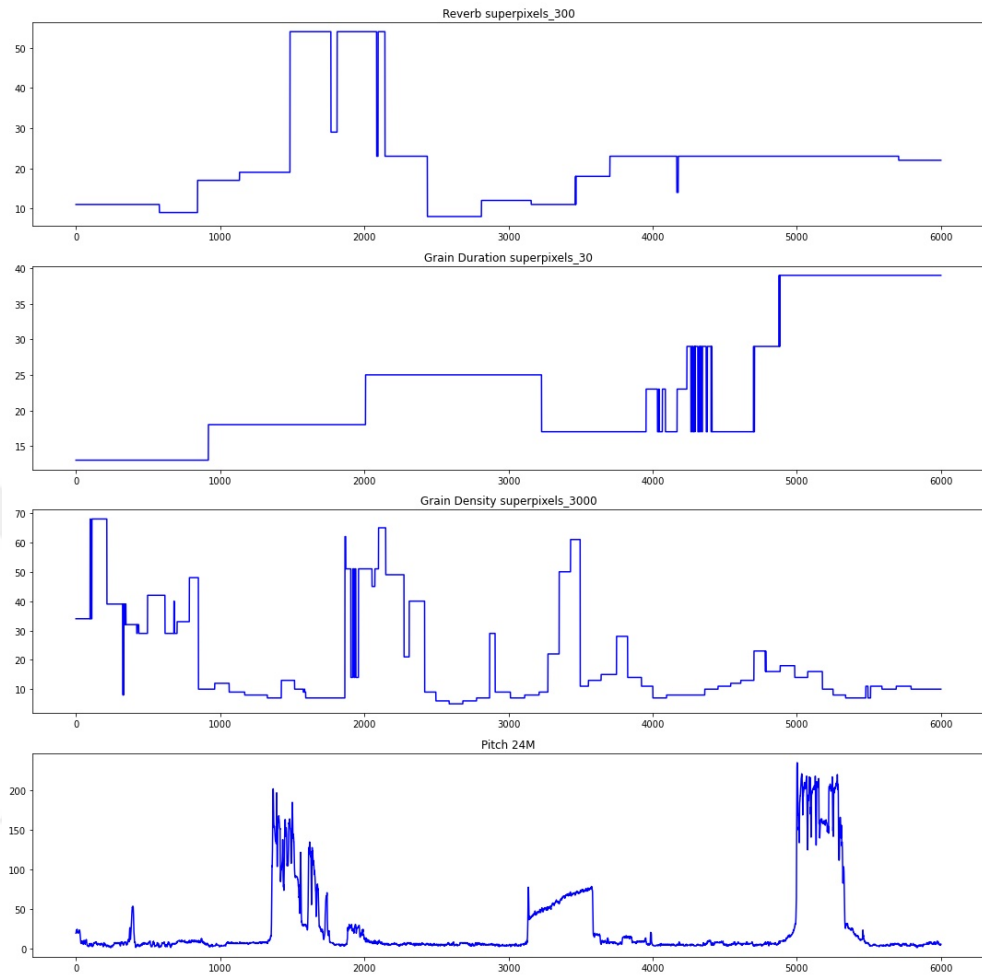
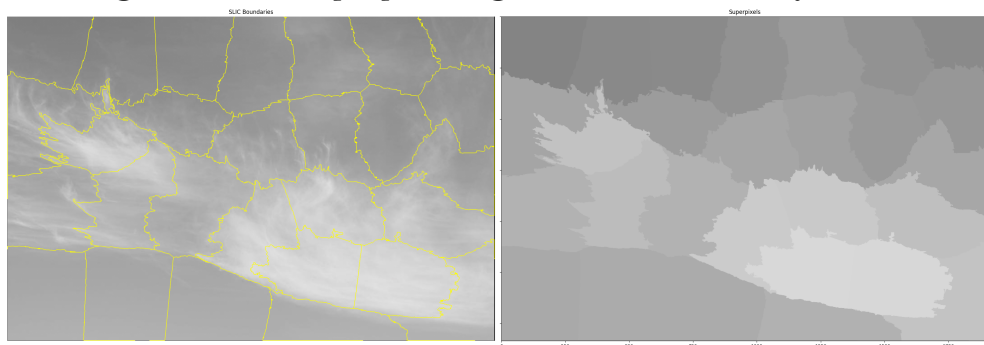


Figure 3.4: 30 superpixel segmentation for Gendy Cloud



Source: Original image captured by the author

```

instr 1
    chnset p4, "kdur"
    chnset p5, "kdens"
    chnset p6, "kpitch"
    chnset p7, "kRevAmount"
endin
instr 2
    igfn = 11
    ibasfrq = 44100 / ftlen(igfn)
    kaoff1 random 0.1, 1
    kaoff2 = 1 - kaoff1
    kpoff1 random ibasfrq * .5, ibasfrq * 5
    kpoff2 = ibasfrq - kpoff1
    imaxgdur = 1
    kdur = 1
    kgdur port kdur, 0.5
    kgdur_p = (round(kgdur*1000))/1000
    kdens = 1
    kdens chnget "kdens"
    kdens_p = round(kdens)
    ibasfrq = 44100 / ftlen(igfn)
    kpitch = 1
    kpitch chnget "kpitch"
    kpitch_p = kpitch
    kamp expseg .001, p3/2, .1, p3/2, .01
    asigL grain kamp, kpitch, kdens, kaoff1, kpoff1, \
            kdur, igfn, 1, imaxgdur, 0.0
    asigR grain kamp, kpitch, kdens, kaoff2, kpoff2, \
            kdur, igfn, 1, imaxgdur, 0.0
    kfblvl = 1
    kfco = 100
    aLRev, aRRev reverbsc asigL, asigR, kfblvl, kfco, sr, 0.5, 1
    kRevAmount = 0.01
    kRevAmount chnget "kRevAmount"
    aGate = 1
    kInstrAmp = 0.2
    outs (asigL + (aLRev*kRevAmount))*kInstrAmp*aGate, \
        (asigR + (aRRev*kRevAmount))*kInstrAmp*aGate
endin

```

Program code 3: The Csound instrument code

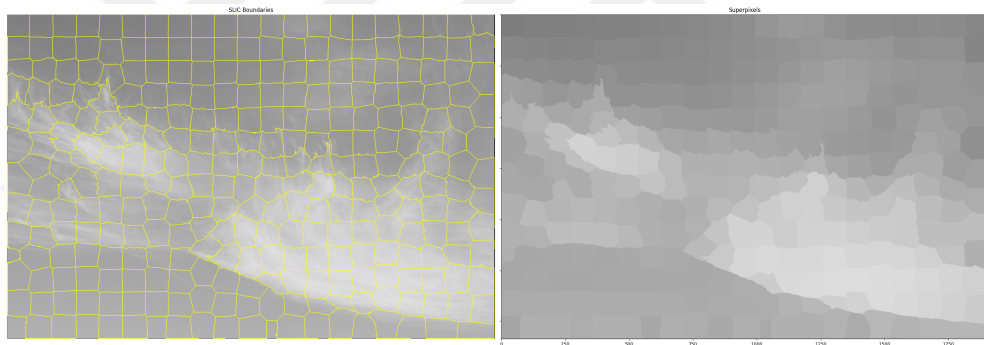
```

; i / start / dur / kdur / kdens / kpitch / kRevAmount
i1 0 0.03 0.01 93 0.78 0.11
i1 0.03 0.03 0.01 93 0.82 0.11
i1 0.06 0.03 0.01 93 0.87 0.11
i1 0.09 0.03 0.01 93 0.95 0.11
i1 0.12 0.03 0.01 93 0.95 0.11
i1 0.15 0.03 0.01 93 0.91 0.11
i1 0.18 0.03 0.01 93 0.95 0.11
i1 0.21 0.03 0.01 93 0.91 0.11
i1 0.24 0.03 0.01 93 1.0 0.11
i1 0.27 0.03 0.01 93 0.87 0.11
i1 0.3 0.03 0.01 93 0.82 0.11
...

```

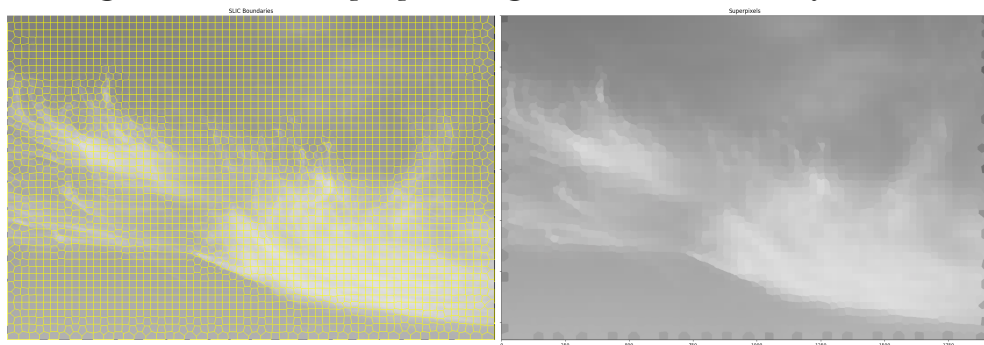
Program code 4: Csound Score sample

Figure 3.5: 300 superpixel segmentation for Gendy Cloud



Source: Original image captured by the author

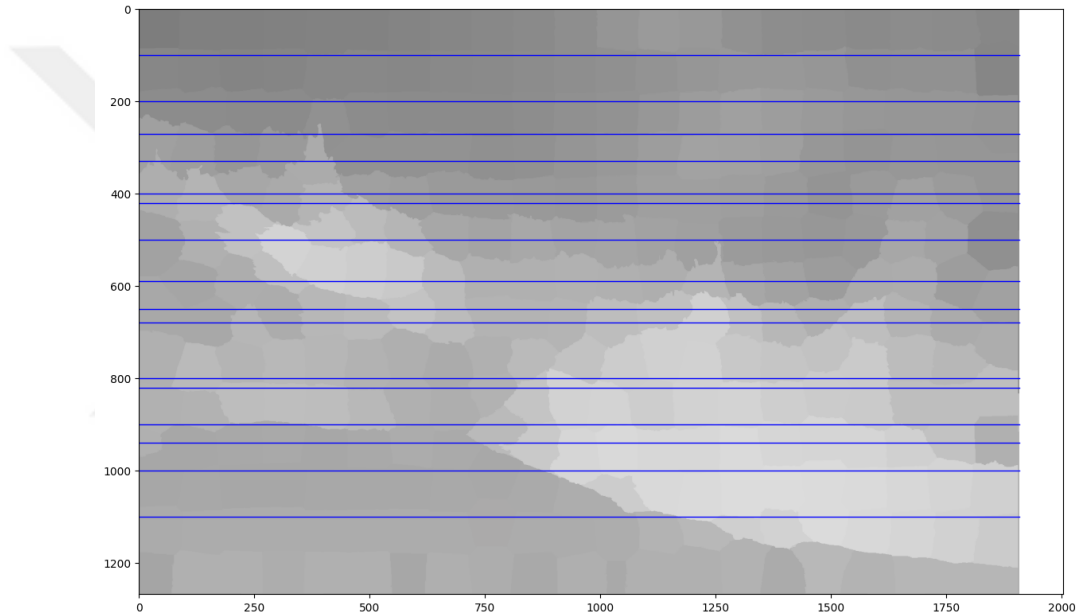
Figure 3.6: 3000 superpixel segmentation for Gendy Cloud



Source: Original image captured by the author

1. On/off switch Deterministic controller which practically mutes and unmutes the sound.
2. Timbre generator Stochastic controller with uniform distribution. That means all internal parameters of Gendy algorithm are determined by a random process whenever the *timbre generator* is triggered.
3. Pattern generator with *note duration* and *rest duration* parameters. Stochastic controller with Poisson distribution. The algorithm would produce values according to durations. Most of the values are accumulated around the given durations but occasionally there could be extreme values.

Figure 3.7: Poisson lambda values assigned to duration and rest values for 8 controller with 300 segment image



Source: Original image captured by the author

Other elements like reverb amount, panning are determined by complete random processes without depending on any external value.

The generated values for each instrument would create some formal structures. Furthermore, we can employ some guides that every instrument should follow. The vertical and horizontal density can be controlled by the pattern generator. Long note durations would end up with drone-like sounds that would create dense vertical textures, whereas short, especially very short sounds below 100 ms, would create granular-like sounds. Rest duration, if chosen as longer values, would build gaps between sound events so that more sparse textures could be viable. Then we can construct higher-level parameters to control the form. (See table 4)

Figure 3.8: Data extracted for eight pattern controller

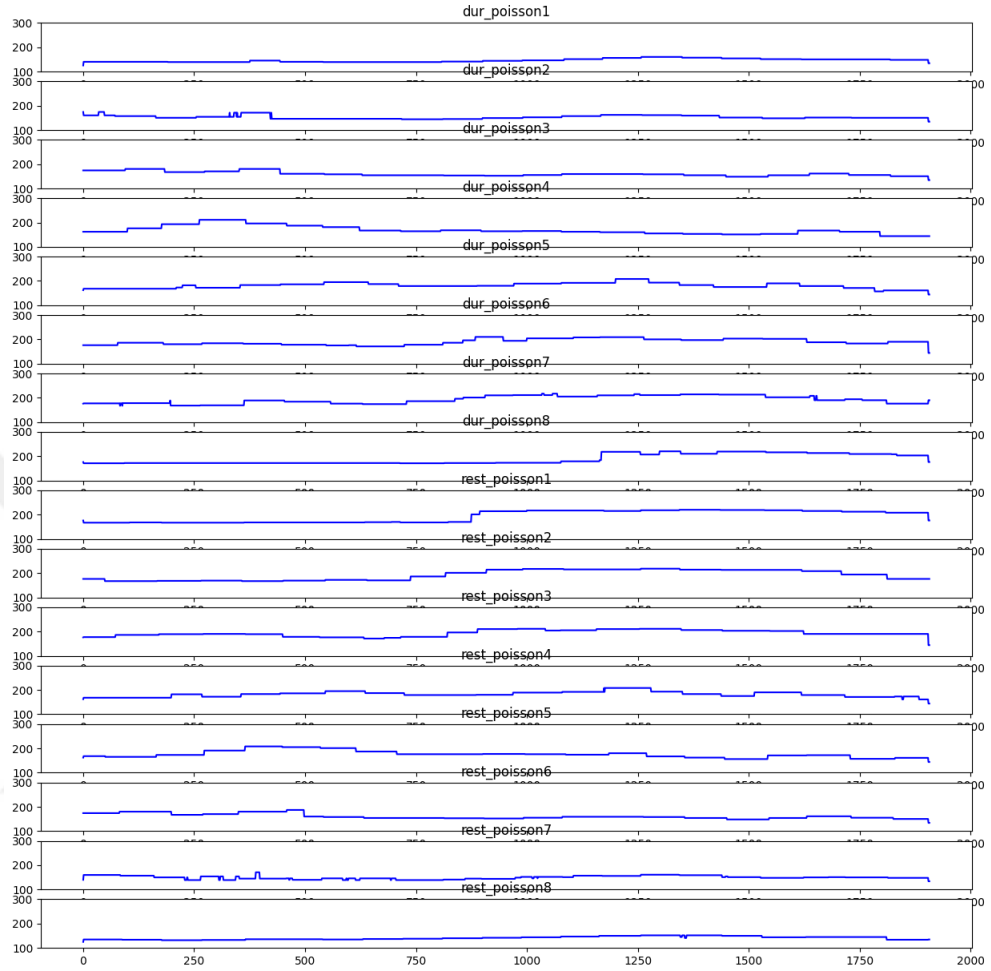


Table 4: Form building with pattern generator

Texture	Note Duration	Rest Duration
Dense	Long	Short
Sparse	Short	Long
Granular	Short	Short
Free-form	Random	Random

```

instr 8 ;; Xenakis gendy
kamp = p4 ;; 0.2 - 0.7
kampdist = p5 ;; 1 - 6
kdurdist = p6 ;; 0.0001 - 1
kadpar = p7 ;; 0.0001 - 1
kddpar = p8 ;; 0.0001 - 1
kminfreq = p9 ;; 20 - 20000
kmaxfreq = p10 ;; 20 - 20000
kampscl = p11 ;; 0.1 - 1
kdurscl = p12 ;; 0.1 - 1
ideg = p13
iuserid = p14
kUserArr[] getrow gkPerfArr, iuserid
kgate = kUserArr[0]
asig gendy kamp, kampdist, kdurdist, kadpar, kddpar,\
      kminfreq, kmaxfreq,kampscl, kdurscl
kenv linsegr 0, 0.01, 1, p3-0.02, 1, 0.01, 0
asig alpass asig, 1, 0.1
ilevel random 0.1, 0.7
asig = asig * kgate
asig = asig*kenv*ilevel
aL, aR pan2 asig, ideg
outs aL, aR
iRvbAmt random 0.4, 0.8
gaReverbL = gaReverbL + (aL * iRvbAmt)
gaReverbR = gaReverbR + (aR * iRvbAmt)
endin

```

Program code 5: Gendy Cloud GENDYN based Csound instrument

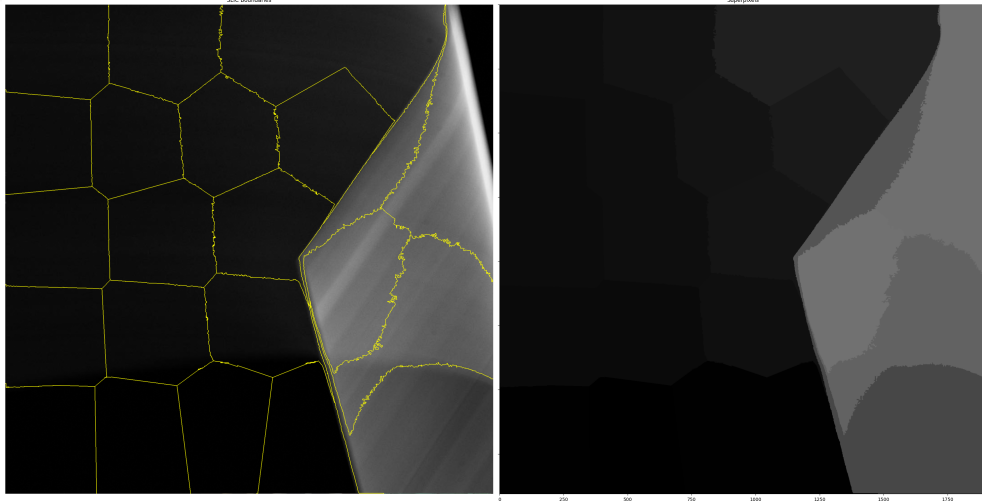
3.3 Mar8

Mar8 is a 8 channel electroacoustic music piece. The sound material is helicopter recordings from home window around Beyoglu area in Istanbul. 15 audio files are extracted after an hour of recording that a helicopter comes into the soundscape and goes. The sound doesn't move circularly in 8-channel diffusion but goes across the listening hall from many directions to eventually build up a thick soundscape above the audience.

The movement of the sound is from one speaker to another with stereo panning. To accomplish the movement eight speakers are paired in the way that sound always crosses the hall. For pairing the first channel is selected randomly, then the channel number is added by four to get the second channel number. Then panning is applied so that the sound moves from one direction to another. The duration of the panning is selected as the same as the duration of the sound.

The sound files are kept in an array and then retrieved by their index number randomly. The sound materials are processed slightly by changing their pitch and time scaling. For a smooth transition *pvtanal*, a phase analysis opcode is used ⁴ Then the analysis is used to resynthesis the sound via *pvsynth* opcode ⁵. A comb filter and an optional resonant filter are used for further processing.

Figure 3.9: 30 superpixel segmentation for Mar8



Source: Original image captured by the author

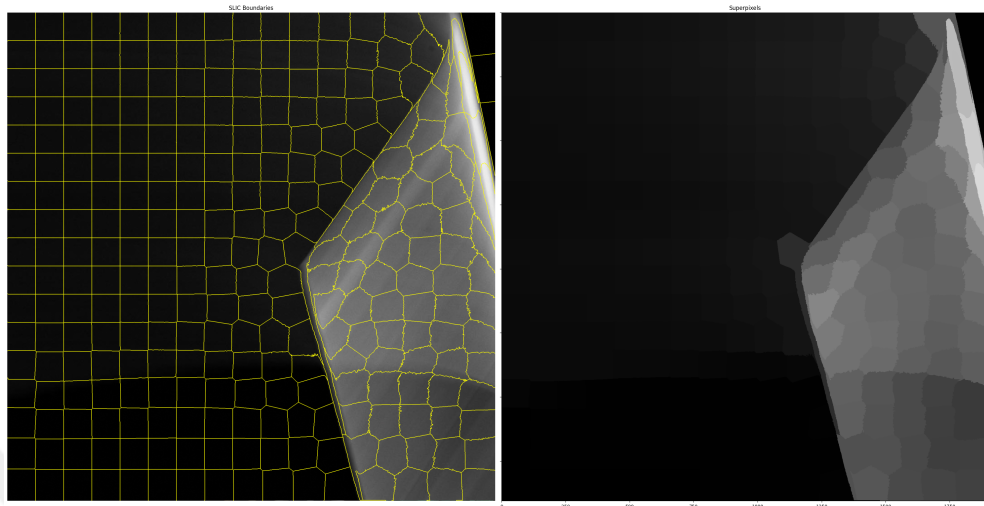
The piece employs a pattern generator that produces values for sound generator instrument. A section generator controls the pattern generator. There three-fold data flow from macro to micro.

section generator -> pattern generator -> sound generator

⁴<https://csound.com/manual/pvtanal.html>

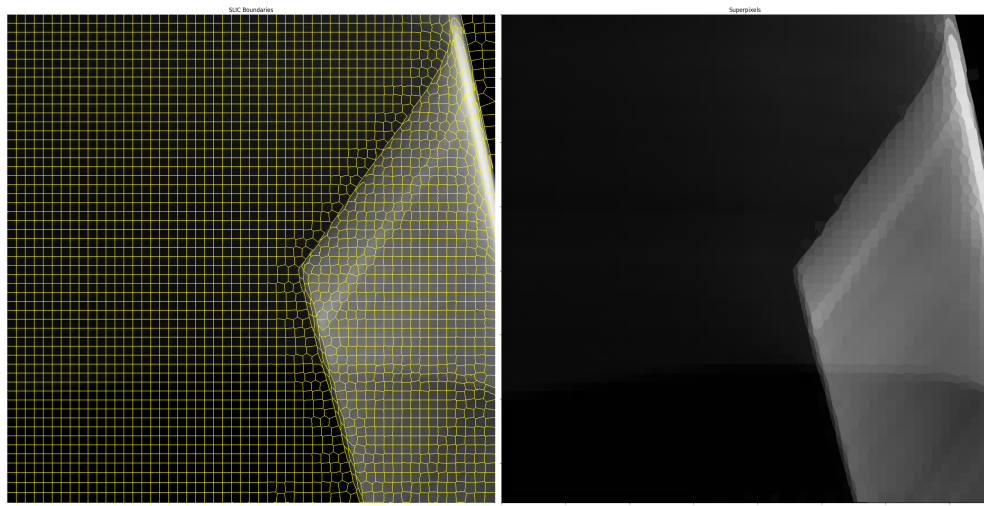
⁵<https://csound.com/manual/pvsynth.html>

Figure 3.10: 300 superpixel segmentation for Mar8



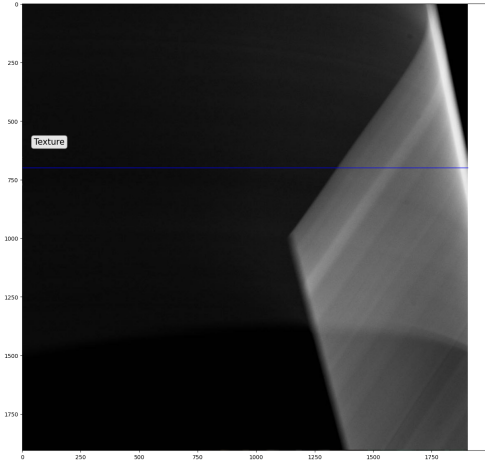
Source: Original image captured by the author

Figure 3.11: 3000 superpixel segmentation for Mar8

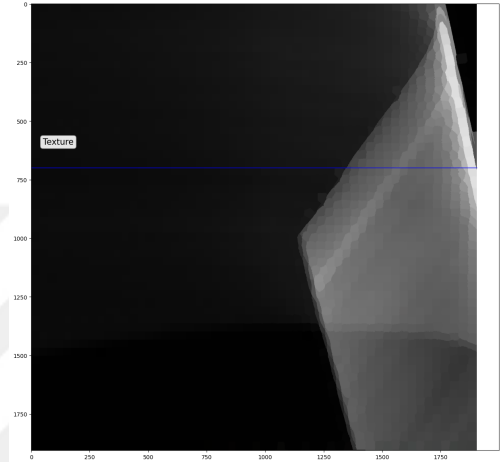


Source: Original image captured by the author

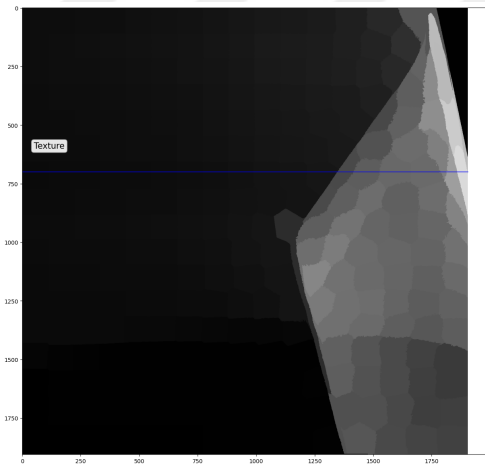
Figure 3.12: Extracting data with several segmentation strategies for *Mar8*



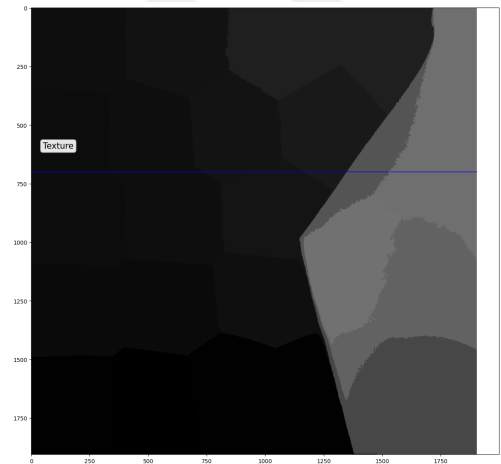
(a) Full resolution image



(b) 3000 segments



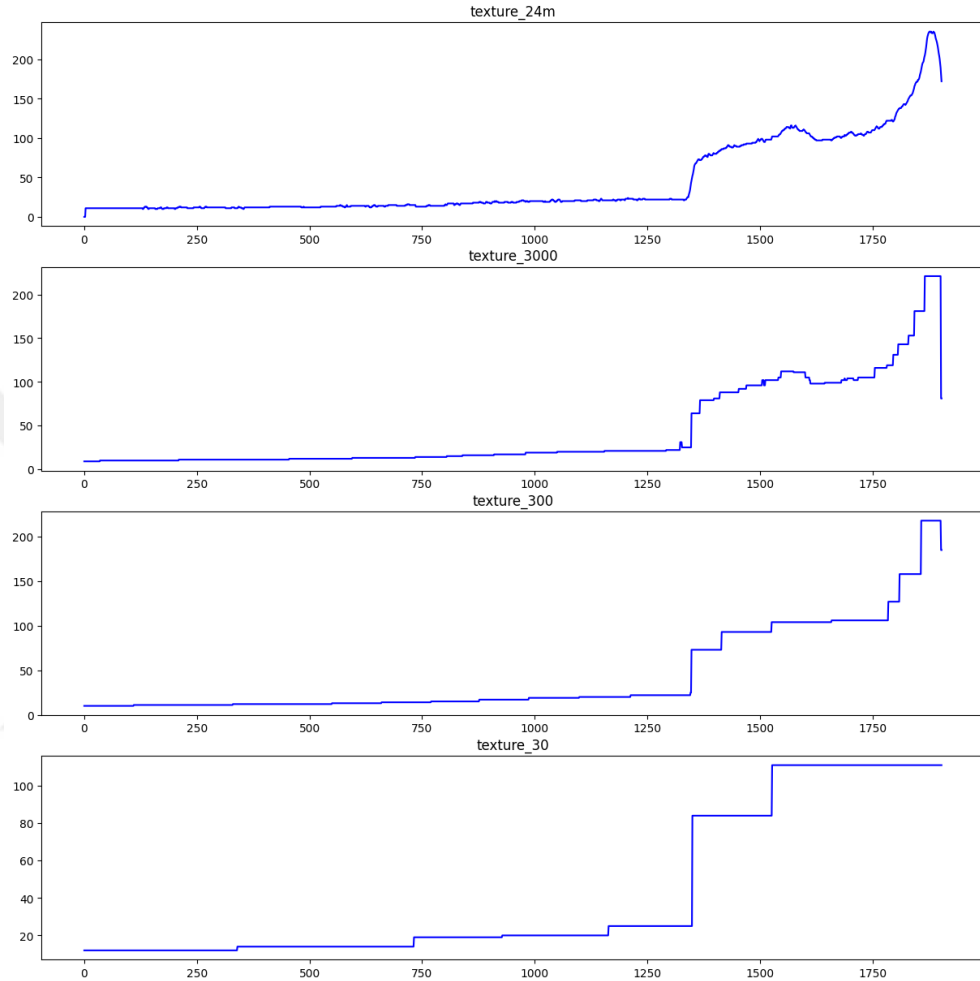
(c) 300 segments



(d) 30 segments

The parameters that are controlled by pattern generator are index of sound file, time-scaling, pitch-shift, two parameters of the resonant filter *areson*⁶, the center frequency of the filter and the bandwidth of the filter, sound movement method and loudness.

Figure 3.13: Visualization of data extracted from the image for Mar8



⁶<https://csound.com/docs/manual/areson.html>

4 CONCLUSIONS

Data sonification as a computational tool for music inherits the relations between the composer, form, and composition process from algorithmic music. Sonification can be achieved by simply mapping data points onto musical parameters. The resultant form, which is emergent, falls into the category of the bottom-up approach. The other approach is top-down, and algorithms that are used for creating musical grammar can be used to define top-level structures and fill them with lower-level sonic activities. The former eliminates the control of the composer on form, and the latter creates a rigid structure to leave no room for formal structures to come out of sonic material. In electroacoustic music musical materials, sound synthesis and processing techniques can derive higher-level structures. The composer of electroacoustic music should benefit from both approaches. This thesis presents a method that can be applied to image sonification to utilize a multiscale form approach (Roads, 2015).

The technical side of this work deals with means of revealing formal structures in data at different levels. The research in computer vision and machine learning provides a solution to generate meaningful segments out of an image that retains the overall look of the image. The esthetic part encounters musical instrument design and mapping strategies. The composer can build higher-level algorithms to map to higher-level data. The instrument represents the micro-compositional features of sound. The pattern and section generator algorithm can be used to control the macro-compositional transformations of structures over time. The composer, architect, and mathematician Iannis Xenakis brings a guideline, "Phases of the Musical Work," that is applicable to our method. His creation of a matrix for his piece *Achorripsis* inspires our solution to the problem of translating two-dimensional data (image) into a linear, time-based art, music. The visual and structural similarity of the *Achorripsis* matrix and the digital image matrix leads to a solution to use one axis for timbral features and the other for a time.

The exemplary music compositions present a practical implementation of the method. The musical algorithms involve a sound generator and higher-level performance tools. Going back and forth between micro and macro musical designs and also in the image segmentation and mapping different levels of parameters to image data proved that a multiscale method is attainable. The overall procedure can be summarized (see 2.6), similar to Xenakis's "Phases."

REFERENCES

- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *Ieee Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2274–2282. <https://doi.org/10.1109/TPAMI.2012.120>
- Adorno, T. W., & Gillespie, S. (1993). Music, Language, and Composition. *The Musical Quarterly*, 77(3), 401–414. <https://doi.org/10.1093/mq/77.3.401>
- Aloimonos, Y., & Rosenfeld, A. (1991). Computer Vision. *Science*, 253(5025), 1249–1254. <https://www.jstor.org/stable/2879170>
- Angeler, D. G., Alvarez-Cobelas, M., & Sánchez-Carrillo, S. (2018). Sonifying social-ecological change: A wetland laments agricultural transformation. *Ecology and Society*, 23(2). <https://doi.org/10.5751/ES-10055-230220>
- Arsenault, L. M. (2002). Iannis Xenakis’s Achorripsis : The Matrix Game. *Computer Music Journal*, 26(1), 58–72. <https://doi.org/10.1162/014892602753633542>
- Attenborough, M. (2003). Probability and statistics. In *Mathematics for Electrical Engineering and Computing* (pp. 493–532). Elsevier. <https://doi.org/10.1016/B978-075065855-3/50047-4>
- Ballora, M., Pennycook, B., Ivanov, P. C., Glass, L., & Goldberger, A. L. (2004). Heart Rate Sonification: A New Approach to Medical Diagnosis. *Leonardo*, 37(1), 41–46. <https://doi.org/10.2307/1577570>
- Barrass, S., & Vickers, P. (2011). Sonification design and aesthetics. In *The Sonification Handbook* (pp. 145–172).
- Ben-Tal, O., & Berger, J. (2004). Creative Aspects of Sonification. *Leonardo*, 37(3), 229–232. <https://doi.org/10.2307/1577727>
- Blessenohl, S., Morrison, C., Criminisi, A., & Shotton, J. (2015). Improving Indoor Mobility of the Visually Impaired With Depth-Based Spatial Sound. 26–34. https://www.cv-foundation.org/openaccess/content_iccv_2015_workshops/w12/html/Blessenohl_Improving_Indoor_Mobility_ICCV_2015_paper.html
- Boulanger, R. (2000). *The Csound book: Perspectives in software synthesis, sound design, signal processing, and programming*. MIT Press.
- Brock, M., & Kristensson, P. O. (2013). Supporting blind navigation using depth sensing and sonification. *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, 255–258. <https://doi.org/10.1145/2494091.2494173>
- Brückner, H.-P., Theimer, W., & Blume, H. (2014). Real-time low latency movement sonification in stroke rehabilitation based on a mobile platform. *2014 IEEE International Conference on Consumer Electronics (ICCE)*, 264–265. <https://doi.org/10.1109/ICCE.2014.6775997>

- Bujacz, M., & Strumillo, P. (2016). Sonification: Review of Auditory Display Solutions in Electronic Travel Aids for the Blind. *Archives of Acoustics*, 401–414. <https://doi.org/10.1515/aoa-2016-0040>
- Caplin, W. E. (2013). *Analyzing classical form: An approach for the classroom*. Oxford University Press.
- Childs, E. (2002). Achorripsis: A sonification of probability distributions. *Proceedings of the 2002 International Conference on Auditory Display, Kyoto, Japan, 2002*.
- Chion, M. (2019). The three listening modes. In *Audio-vision: Sound on screen* (pp. 22–34). Columbia University Press. <https://doi.org/10.7312/chio18588-004>
- Collins, N. (2009). Musical Form and Algorithmic Composition. *Contemporary Music Review*, 28(1), 103–114. <https://doi.org/10.1080/07494460802664064>
- Copley, P., & Gartland-Jones, A. (2005). Musical form and algorithmic solutions. *Proceedings of the 5th Conference on Creativity & Cognition*, 226–231. <https://doi.org/10.1145/1056224.1056258>
- Delatour, T. (2000). Molecular Music: The Acoustic Conversion of Molecular Vibrational Spectra. *Computer Music Journal*, 24(3), 48–68. <https://doi.org/10.1162/014892600559335>
- Diaz-Merced, W. L., Candey, R. M., Brickhouse, N., Schneps, M., Mannone, J. C., Brewster, S., & Kolenberg, K. (2011). Sonification of Astronomical Data. *Proceedings of the International Astronomical Union*, 7(S285), 133–136. <https://doi.org/10.1017/S1743921312000440>
- Dubus, G., & Bresin, R. (2013). A Systematic Review of Mapping Strategies for the Sonification of Physical Quantities. *Plos One*, 8(12), e82491. <https://doi.org/10.1371/journal.pone.0082491>
- Dubus, G., & Bresin, R. (2015). Exploration and evaluation of a system for interactive sonification of elite rowing. *Sports Engineering*, 18(1), 29–41. <https://doi.org/10.1007/s12283-014-0164-0>
- Effenberg, A., Fehse, U., & Weber, A. (2011). Movement Sonification: Audiovisual benefits on motor learning. *Bio Web of Conferences*, 1, 00022. <https://doi.org/10.1051/bioconf/20110100022>
- Embarak, D. O. (2018). *Data Analysis and Visualization Using Python: Analyze Data to Create Visualizations for BI Systems*. Apress. <https://doi.org/10.1007/978-1-4842-4109-7>
- George, S. S., Crawford, D., Reubold, T., & Giorgi, E. (2017). Making Climate Data Sing: Using Music-like Sonifications to Convey a Key Climate Record. *Bulletin of the American Meteorological Society*, 98(1), 23–27. <https://doi.org/10.1175/BAMS-D-15-00223.1>
- Gibson, J. (2006). sLowlife: Sonification of Plant Study Data. *Leonardo Music Journal*, 16, 42–44. <https://doi.org/10.1162/lmj.2006.16.42b>

- Graham, S., & Simons, J. (2021). Listening to Dura Europos: An Experiment in Archaeological Image Sonification. *Internet Archaeology*, 56. <https://doi.org/10.11141/ia.56.8>
- Grond, F., & Hermann, T. (2012). Aesthetic strategies in sonification. *Ai & Society*, 27(2), 213–222. <https://doi.org/10.1007/s00146-011-0341-7>
- Hansen, C. D., & Johnson, C. R. (2005). *The Visualization Handbook* (1st ed.). Elsevier-Butterworth Heinemann.
- Hedelin, F. (2008). Formalising Form: An Alternative Approach To Algorithmic Composition. *Organised Sound*, 13(3), 249–257. <https://doi.org/10.1017/S1355771808000344>
- Hermann, T. (2008). *Taxonomy and Definitions for Sonification and Auditory Display*.
- Hermann, T. (2011). Model-Based Sonification. In T. Hermann, A. Hunt, & J. G. Neuhoff (Eds.), *The Sonification Handbook*. Logos Verlag.
- Hoffmann, P. (2000). The New GENDYN Program. *Computer Music Journal*, 24(2), 31–38. <http://www.jstor.org/stable/3681925>
- Hunt, A., & Wanderley, M. M. (2002). Mapping performer parameters to synthesis engines. *Organised Sound*, 7(2), 97–108. <https://doi.org/10.1017/S1355771802002030>
- Ikeda, A., Tanaka, Y., Hwang, D.-H., Kon, H., & Koike, H. (2019). Golf training system using sonification and virtual shadow. *ACM SIGGRAPH 2019 Emerging Technologies*, 1–2. <https://doi.org/10.1145/3305367.3327993>
- Jensen, M. (2008). Composing DNA Music Within the Aesthetics of Chance. *Perspectives of New Music*, 46(2), 243–259. <https://www.jstor.org/stable/25652395>
- Jensen, M. G. (2009). John Cage, Chance Operations, and the Chaos Game: Cage and the “I Ching”. *The Musical Times*, 150(1907), 97–102. <https://doi.org/10.2307/25597623>
- Kabisch, E., Kuester, F., & Penny, S. (2005). Sonic panoramas: Experiments with interactive landscape image sonification. *Proceedings of the 2005 International Conference on Augmented Tele-Existence*, 156–163. <https://doi.org/10.1145/1152399.1152428>
- Kayukawa, S., Ishihara, T., Takagi, H., Morishima, S., & Asakawa, C. (2020). Guiding Blind Pedestrians in Public Spaces by Understanding Walking Behavior of Nearby Pedestrians. *Proceedings of the Acm on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3), 85:1–85:22. <https://doi.org/10.1145/3411825>
- Kirke, A., Freeman, S., & Miranda, E. R. (2015). Wireless Interactive Sonification of Large Water Waves to Demonstrate the Facilities of a Large-Scale Research Wave Tank. *Computer Music Journal*, 39(3), 59–70. https://doi.org/10.1162/COMJ_a_00315
- Kramer, G., & Walker, B. N. (2005). Sound science: Marking ten international

- conferences on auditory display. *Acm Transactions on Applied Perception*, 2(4), 383–388. <https://doi.org/10.1145/1101530.1101531>
- Kramer, G., Walker, B., Bonebright, T., Cook, P., Flowers, J., Miner, N., Neuhoff, J., Bargar, R., Barrass, S., Berger, J., Evreinov, G., Fitch, W., Grohn, M., Handel, S., Kaper, H., Levkowitz, H., Lodha, S., Shinn-Cunningham, B., Simoni, M., & Tipei, S. (1999). *The sonification report: Status of the field and research agenda. Report prepared for the National Science Foundation by members of the International Community for Auditory Display.*
- Manning, P. (2013). *Electronic and computer music* (4th ed). Oxford University Press.
- Martinez, W. L. M. A. R. (2004). *Exploratory Data Analysis with MATLAB* (1st ed.).
- McGee, R. (2013). VOSIS: A multi-touch image sonification interface. *Proceedings of the International Conference on New Interfaces for Musical Expression*, 460–463. <https://doi.org/10.5281/zenodo.1178604>
- Miranda, E. R. (2001). *Composing music with computers* (1st ed). Focal Press.
- Miranda, E. R., Bull, L., Gueguen, F., & Uroukov, I. S. (2009). Computer Music Meets Unconventional Computing: Towards Sound Synthesis with in vitro Neuronal Networks. *Computer Music Journal*, 33(1), 9–18. <https://doi.org/0.1162/comj.2009.33.1.9>
- Narmour, E. (1991). The Top-down and Bottom-up Systems of Musical Implication: Building on Meyer’s Theory of Emotional Syntax. *Music Perception*, 9(1), 1–26. <https://doi.org/10.2307/40286156>
- Nierhaus, G. (2009). *Algorithmic Composition*. Springer. <https://doi.org/10.1007/978-3-211-75540-2>
- O’Brien, B., Vidal, A., Bringoux, L., & Bourdin, C. (2019, November 18). Developing Movement Sonification for Sports Performance: A survey of studies developed at the Institute of Movement Science. *Proceedings of ISON 2019, 6th Interactive Sonification Workshop*. <https://doi.org/10.5281/zenodo.3756793>
- Pauletto, S., & Hunt, A. (2009). Interactive sonification of complex data. *International Journal of Human-Computer Studies*, 67(11), 923–933. <https://doi.org/10.1016/j.ijhcs.2009.05.006>
- Pennebaker, W. B., & Mitchell, J. L. (1992). *JPEG still image data compression standard*. Van Nostrand Reinhold.
- Pietschmann, J., Geu Flores, F., & Jöllenbeck, T. (2019). Gait Training in Orthopedic Rehabilitation after Joint Replacement - Back to Normal Gait with Sonification? *International Journal of Computer Science in Sport*, 18(2), 34–48. <https://doi.org/10.2478/ijcss-2019-0012>
- Polli, A. (2005). “Atmospherics/Weather Works”: A Spatialized Meteorological Data Sonification Project. *Leonardo*, 38(1), 31–36. <https://doi.org/10.1162/leon.2005.38.1.31>

- Polo, A., & Sevillano, X. (2019). Musical Vision: An interactive bio-inspired sonification tool to convert images into music. *Journal on Multimodal User Interfaces*, 13(3), 231–243. <https://doi.org/10.1007/s12193-018-0280-4>
- Rector, K., Bennett, C. L., & Kientz, J. A. (2013). Eyes-free yoga: An exergame using depth cameras for blind & low vision exercise. *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, 1–8. <https://doi.org/10.1145/2513383.2513392>
- Ren, & Malik. (2003). Learning a classification model for segmentation. *Proceedings Ninth IEEE International Conference on Computer Vision*, 10–17 vol.1. <https://doi.org/10.1109/ICCV.2003.1238308>
- Ridley, A. L., & Birchall, C. (2020). Evaluating data visualization: Broadening the measurements of success. In M. Engebretsen & H. Kennedy (Eds.), *Data Visualization in Society* (pp. 127–140). Amsterdam University Press. <https://doi.org/10.2307/j.ctvzgb8c7.14>
- Roads, C. (2001). *Microsound*. MIT Press.
- Roads, C. (2015). *Composing electronic music: A new aesthetic*. Oxford University Press.
- Salter, C. L., Baalman, M. A. J., & Moody-Grigsby, D. (2008). Between Mapping, Sonification and Composition: Responsive Audio Environments in Live Performance. In R. Kronland-Martinet, S. Ystad, & K. Jensen (Eds.), *Computer Music Modeling and Retrieval. Sense of Sounds* (Vol. 4969, pp. 246–262). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-85035-9_17
- Sarkar, R., Bakshi, S., & Sa, P. K. (2012). Review on image sonification: A non-visual scene representation. *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, 86–90. <https://doi.org/10.1109/RAIT.2012.6194485>
- Scaletti, C. (1994). Chapter 8: Sound synthesis algorithms for auditory data representations. pp. 223–251. In G. Kramer (Ed.), *Auditory display: Sonification, audification, and auditory interfaces*. Addison-Wesley.
- Scaletti, C. (2018). Sonification Music. In R. T. Dean & A. McLean (Eds.), *The Oxford Handbook of Algorithmic Music* (Vol. 1). Oxford University Press. <https://doi.org/10.1093/oxfordhb/9780190226992.013.9>
- Schaffert, N., & Mattes, K. (2015). Interactive Sonification in Rowing: Acoustic Feedback for On-Water Training. *Ieee Multimedia*, 22(1), 58–67. <https://doi.org/10.1109/MMUL.2015.9>
- Schaffert, N., Godbout, A., Schlueter, S., & Mattes, K. (2017). Towards an application of interactive sonification for the forces applied on the pedals during cycling on the Wattbike ergometer. *Displays*, 50, 41–48. <https://doi.org/10.1016/j.displa.2017.09.004>
- Schaffert, N., Janzen, T. B., Mattes, K., & Thaut, M. H. (2019). A Review on the Relationship Between Sound and Movement in Sports and Rehabilitation. *Frontiers in Psychology*, 10. <https://www.frontiersin.org/articles/10.>

- Scholz, D. S., Wu, L., Pirzer, J., Schneider, J., Rollnik, J. D., Großbach, M., & Altenmüller, E. O. (2014). Sonification as a possible stroke rehabilitation strategy. *Frontiers in Neuroscience*, 8. <https://www.frontiersin.org/articles/10.3389/fnins.2014.00332>
- Schönberg, A. (1999). *Fundamentals of musical composition* (G. Strang & L. Stein, Eds.). faber and faber.
- Seibert, G., & Hug, D. (2013). Bringing musicality to movement sonification: Design and evaluation of an auditory swimming coach. *Proceedings of the 8th Audio Mostly Conference*, 1–6. <https://doi.org/10.1145/2544114.2544127>
- Shrewsbury, B. T. (2011). Providing haptic feedback using the kinect. *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*, 321–322. <https://doi.org/10.1145/2049536.2049628>
- Sigrist, R., Fox, S., Riener, R., & Wolf, P. (2016). Benefits of Crank Moment Sonification in Cycling. *Procedia Engineering*, 147, 513–518. <https://doi.org/10.1016/j.proeng.2016.06.230>
- Sigrist, R., Rauter, G., Marchal-Crespo, L., Riener, R., & Wolf, P. (2015). Sonification and haptic feedback in addition to visual feedback enhances complex motor task learning. *Experimental Brain Research*, 233(3), 909–925. <https://doi.org/10.1007/s00221-014-4167-7>
- Squibbs, R., Solomos, M., Georgaki, A., Zervos, G., & Xenakis, I. (n.d.). *The Composer's Flair: Achorripsis as Music*. 8.
- Stienstra, J., Overbeeke, K., & Wensveen, S. (2011). Embodying complexity through movement sonification: Case study on empowering the speed-skater. *Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer-Human Interaction: Facing Complexity*, 39–44. <https://doi.org/10.1145/2037296.2037310>
- Sturm, B. L. (2005). Pulse of an Ocean: Sonification of Ocean Buoy Data. *Leonardo*, 38(2), 143–149. <https://doi.org/10.1162/0024094053722453>
- Supper, M. (2001). A Few Remarks on Algorithmic Composition. *Computer Music Journal*, 25(1), 48–53. <http://www.jstor.org/stable/3681634>
- Tanaka, Y., Kon, H., & Koike, H. (2018). A real-time golf-swing training system using sonification and sound image localization. *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, 1–2. <https://doi.org/10.1145/3281505.3281604>
- Varèse, E., & Wen-chung, C. (1966). The Liberation of Sound. *Perspectives of New Music*, 5(1), 11–19. <https://doi.org/10.2307/832385>
- Vickers, P., & Hogg, B. (2013, November 21). *Sonification Abstraite/Sonification Concrete: An 'Aesthetic Perspective Space' for Classifying Auditory Displays in the Ars Musica Domain*. <https://doi.org/10.48550/arXiv.1311.5426>

- Väljamäe, A., Steffert, T., Holland, S., Marimon, X., Benitez, R., Mealla, S., Oliveira, A., & Jordà, S. (2013). *A review of real-time EEG sonification research*. 85–93. <http://icad2013.com/index.php>
- Waldrop, M. M. (1984). Computer Vision. *Science*, 224(4654), 1225–1227. <http://www.jstor.org/stable/1692050>
- Weinberg, G., & Thatcher, T. (2006). Interactive Sonification: Aesthetics, Functionality and Performance. *Leonardo Music Journal*, 16, 9–12. <https://doi.org/10.1162/lmj.2006.16.9>
- Wickham, H., & Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data* (1st ed.). O'Reilly Media.
- William W. Gaver. (1989). The SonicFinder: An interface that uses auditory icons. *Human-Computer Interaction*, 4(1), 67–94. https://doi.org/10.1207/s15327051hci0401_3
- Xenakis, I. (1992). *Formalized music: Thought and mathematics in composition* (S. Kanach, Tran.; Rev. ed). Pendragon Press.
- Xenakis, I., & Varga, B. A. (1996). *Conversations with Iannis Xenakis*. Faber and Faber.
- Young, J. (2016). Forming form. In S. Emmerson & L. Landy (Eds.), *Expanding the Horizon of Electroacoustic Music Analysis* (pp. 58–79). Cambridge University Press. <https://doi.org/10.1017/CB09781316339633.004>
- van Rheden, V., Grah, T., & Meschtscherjakov, A. (2020). Sonification approaches in sports in the past decade: A literature review. *Proceedings of the 15th International Audio Mostly Conference*, 199–205. <https://doi.org/10.1145/3411109.3411126>
- van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., & Yu, T. (2014). Scikit-image: Image processing in Python. *Peerj*, 2, e453. <https://doi.org/10.7717/peerj.453>

A SOURCE CODES

A.1 Granular Cello Python Code

```
import numpy as np
from skimage import data, io
from skimage import color
from skimage import morphology
from skimage import segmentation
from skimage.segmentation import find_boundaries
from matplotlib import pyplot as plt

def segment_slic(img, n_segm):
    # SLIC result
    slic = segmentation.slic(img, n_segments=n_segm, start_label=1)
    b = find_boundaries(slic)
    superpixels = color.label2rgb(slic, img, kind='avg', bg_label=0)

    # Display result
    fig, ax_arr = plt.subplots(1, 2, sharex=True, sharey=True,
        ↪ figsize=(30, 30))
    ax1, ax2 = ax_arr
    ax1.imshow(segmentation.mark_boundaries(img, slic))
    ax1.set_title("SLIC Boundaries")

    ax2.imshow(superpixels)
    ax2.set_title("Superpixels")

    for ax in ax_arr.ravel():
        ax.set_axis_off()
        plt.tight_layout()
        plt.show()

    return [slic, superpixels]

def param_arr(data, param):
    loc = params[param]["y_loc"]
    return data[loc,:][:,0]

def plot_params_on_image(data, params):
    fig, ax_arr = plt.subplots(1, 1, sharex=True, sharey=True,
        ↪ figsize=(15, 15))
    ax1 = ax_arr
    ax1.imshow(data)
    for key in params:
        x1 = 0
        x2 = np.size(data[0,:])/3
        y1 = params[key]["y_loc"]
        y2 = y1
        ax1.plot([x1, x2], [y1, y2], color='b', linestyle='-',
            ↪ linewidth=1)
```

```

bbox_props = dict(boxstyle="round", fc="w", ec="0.5", alpha=0.9)
ax1.annotate(params[key]["name"], xy=(50, params[key]["y_loc"] -
↪ 100), xycoords='data', bbox=bbox_props, size=15)

img = io.imread('./images/DSC03969.JPG')
segments_3000, superpixels_3000 = segment_slic(img, 3000)
segments_300, superpixels_300 = segment_slic(img, 300)
segments_30, superpixels_30 = segment_slic(img, 30)

params = {"reverb": {"name": "Reverb", "y_loc": 700, "minmax": [0.1,
↪ 2.5]},
          "gdens": {"name": "Grain Density", "y_loc": 1400, "minmax":
↪ [0.1, 2.5]},
          "gdur": {"name": "Grain Durations", "y_loc": 2050, "minmax":
↪ [0.1, 2.5]},
          "gpitch": {"name": "Grain Pitch", "y_loc": 2750, "minmax":
↪ [0.1, 2.5]},
          }

# DENS Plots
gdens_24m = param_arr(img, "gdens")
gdens_3000 = param_arr(superpixels_3000, "gdens")
gdens_300 = param_arr(superpixels_300, "gdens")
gdens_30 = param_arr(superpixels_30, "gdens")

fig, axs = plt.subplots(4, figsize=(15,15))
fig.suptitle('')

axs[0].plot(gdens_24m, color='b', label='gdens')
axs[0].title.set_text('gdens_24m')
axs[0].set_ylim([0, 90])

axs[1].plot(gdens_3000, color='b', label='gdens')
axs[1].title.set_text('gdens_3000')
axs[1].set_ylim([0, 90])

axs[2].plot(gdens_300, color='b', label='gdens')
axs[2].title.set_text('gdens_300')
axs[2].set_ylim([0, 90])

axs[3].plot(gdens_30, color='b', label='gdens')
axs[3].title.set_text('gdens_30')
axs[3].set_ylim([0, 90])

reverb_data = param_arr(superpixels_300, "reverb")
gdur_data = param_arr(superpixels_30, "gdur")
gpitch_data = param_arr(img, "gpitch")
gdens_data = param_arr(superpixels_3000, "gdens")

fig, axs = plt.subplots(4, figsize=(15,15))

```

```

fig.suptitle('')
axs[0].plot(reverb_data, color='b', label='Reverb')
axs[0].title.set_text('Reverb superpixels_300')

axs[1].plot(gdur_data, color='b', label='Dur')
axs[1].title.set_text('Grain Duration superpixels_30')

axs[2].plot(gdens_data, color='b', label='Dens')
axs[2].title.set_text('Grain Density superpixels_3000')

axs[3].plot(gpitch_data, color='b', label='Pitch')
axs[3].title.set_text('Pitch 24M')
fig.tight_layout()

# Score generation for Csound Granular Synth
# igfn = p4
# kdur = p5
# kdens = p6
# kpitchRatio = p7
# kRevAmount = p8

gdur_data = np.interp(gdur_data, (gdur_data.min(), gdur_data.max()),
    ↪ (0.01, 1.5))
gdens_data = np.interp(gdens_data, (gdens_data.min(), gdens_data.max()),
    ↪ (1, 200))
gpitch_data = np.interp(gpitch_data, (gpitch_data.min(),
    ↪ gpitch_data.max()), (0.01, 3))
reverb_data = np.interp(reverb_data, (reverb_data.min(),
    ↪ reverb_data.max()), (0.1, 0.2))

total_dur = 3 * 60
start = 0
dur = total_dur/np.size(gdur_data)
filename =
    ↪ f'/Users/serkansevilgen/_YTU/Hisarli-presentation/code/granular.sco'
f = open(filename, 'a')
instr = 1
print(f";i | start | dur | kdur | kdens | kpitch | kRevAmount ",
    ↪ file=f)
for i in range(np.size(gdur_data)):
    kdur = round(gdur_data[i], 2)
    kdens = round(gdens_data[i])
    kpitch = round(gpitch_data[i], 2)
    kRevAmount = round(reverb_data[i], 2)
    print(f";i{instr} {start} {dur} {kdur} {kdens} {kpitch}
        ↪ {kRevAmount}", file=f)
    start += dur
    start = round(start, 2)

```

A.2 Mar8 Csound Code

```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 32
Odbfs = 1
nchnls = 8

;; PATCHBAY for differenet speaker setups

;; 8 channels Setup cross
;; giOutChn_1 = 1
;; giOutChn_2 = 2
;; giOutChn_3 = 4
;; giOutChn_4 = 6
;; giOutChn_5 = 8
;; giOutChn_6 = 7
;; giOutChn_7 = 5
;; giOutChn_8 = 3

;; 8 channels Setup octagone
giOutChn_1 = 1
giOutChn_2 = 2
giOutChn_3 = 3
giOutChn_4 = 4
giOutChn_5 = 5
giOutChn_6 = 6
giOutChn_7 = 7
giOutChn_8 = 8

;; Stereo mix
;; giOutChn_1 = 1
;; giOutChn_2 = 2
;; giOutChn_8 = 1
;; giOutChn_3 = 2
;; giOutChn_7 = 1
;; giOutChn_4 = 2
;; giOutChn_6 = 1
;; giOutChn_5 = 2

giOut[] init 9
giOut[] fillarray 0, giOutChn_1, giOutChn_2, giOutChn_3, giOutChn_4,
↪ giOutChn_5, giOutChn_6, giOutChn_7, giOutChn_8

;; gaMix[] init 9
;; gaMix[0] = 0
;; gaMix[1] = 0
;; gaMix[2] = 0
```

```

;; gaMix[3] = 0
;; gaMix[4] = 0
;; gaMix[5] = 0
;; gaMix[6] = 0
;; gaMix[7] = 0
;; gaMix[8] = 0

seed 21

instr stop
  iins = p4
  imode = p5
  irel = 1
  ;; imode
  ;; turn off all instances (0), oldest only (1), or newest only (2)
  ;; 4: only turn off notes with exactly matching (fractional)
  ↪ instrument number
  ;; 8: only turn off notes with indefinite duration (p3 < 0 or MIDI)
  turnoff2 iins, imode, irel
endin

giHeli0 ftgen 0, 0, 0, 1, "materials/helicopter-001.wav", 0, 0, 0
giHeli1 ftgen 0, 0, 0, 1, "materials/helicopter-002.wav", 0, 0, 0
giHeli2 ftgen 0, 0, 0, 1, "materials/helicopter-003.wav", 0, 0, 0
giHeli3 ftgen 0, 0, 0, 1, "materials/helicopter-004.wav", 0, 0, 0
giHeli4 ftgen 0, 0, 0, 1, "materials/helicopter-005.wav", 0, 0, 0
giHeli5 ftgen 0, 0, 0, 1, "materials/helicopter-006.wav", 0, 0, 0
giHeli6 ftgen 0, 0, 0, 1, "materials/helicopter-007.wav", 0, 0, 0
giHeli7 ftgen 0, 0, 0, 1, "materials/helicopter-008.wav", 0, 0, 0
giHeli8 ftgen 0, 0, 0, 1, "materials/helicopter-009.wav", 0, 0, 0
giHeli9 ftgen 0, 0, 0, 1, "materials/helicopter-010.wav", 0, 0, 0
giHeli10 ftgen 0, 0, 0, 1, "materials/helicopter-011.wav", 0, 0, 0
giHeli11 ftgen 0, 0, 0, 1, "materials/helicopter-012.wav", 0, 0, 0
giHeli12 ftgen 0, 0, 0, 1, "materials/helicopter-013.wav", 0, 0, 0
giHeli13 ftgen 0, 0, 0, 1, "materials/helicopter-014.wav", 0, 0, 0
giHeli14 ftgen 0, 0, 0, 1, "materials/helicopter-015.wav", 0, 0, 0

giHeliArr[] init 15
giHeliArr[] fillarray giHeli0, giHeli1, giHeli2, giHeli3, giHeli4,
  ↪ giHeli5, giHeli6, giHeli7, giHeli8, giHeli9, giHeli10, giHeli11,
  ↪ giHeli12, giHeli13, giHeli14

instr 3 ;; Women's March Istanbul helicopters 15 audio
  ifndx = p4 ; 0 - 14
  ifn = giHeliArr[ifndx]
  kfn = ifn
  ;; p3 = nsamp(fn)/sr
  kdetect init 0
  kwrap init 0
  kamp init 1

```

```

ktimescal init p5
kpitch init p6
kcf init p7
kbw init p8
imov init p9
klevel = p10
ilen = ftlen(ifn)/sr
itimescal = i(ktimescal)
;; TODO why p3 is half of the actual length
p3 = ilen/itimescal
fsig pvstanal ktimescal, kamp, kpitch, kfn, kdetect, kwrap
asig pvsynth fsig
krvt = 0.1
    asig comb asig, krvt, .1
    asig comb asig, krvt, .1
    asig comb asig, krvt, .1
    asig comb asig, krvt, .1

    if (kcf != 0) then
        afilt areson asig, kcf, kbw
        asig balance afilt, asig
    endif
asig alpass asig, 0.8, 0.1

;; PANNING
asig = asig *klevel
if (imov == 0) then
    kpan line -1.0, p3*2, 1.0
    asig1, asig2 pan2 asig, kpan
    ichnl1 random 1, 8
    ichnl1 = int(ichnl1)
    ichnl2 = ichnl1 + 4
    ichnl2 = ichnl2 % 9
    outch giOut[ichnl1], asig1, giOut[ichnl2], asig2
    ;; gaMix[ichnl1] = asig1
    ;; gaMix[ichnl2] = asig2
elseif (imov == 1) then
    kcps = p3*2
    k1 phasor kcps ; "fraction" of circle -
    ↪ controls speed of rotation - can be negative
    k2 tablei k1, 1, 1 ; sin of angle (sinusoid in
    ↪ f1)
    k3 tablei k1, 1, 1, .25, 1 ; cos of angle (sin offset
    ↪ 1/4 circle)
    asig1, asig2, asig3, asig4 pan asig, k2/2, k3/2, 2, 1, 1 ;
    ↪ sent in a circle (f2=1st quad sin)
    ichnl1 random 1, 8
    ;; TODO: the below needs a loop
    ichnl1 = int(ichnl1)
    ichnl2 = ichnl1 + 1
    ichnl2 = ichnl2 % 9

```

```

        ichnl3 = ichnl1 + 2
        ichnl3 = ichnl3 % 9
        ichnl4 = ichnl1 + 3
        ichnl4 = ichnl4 % 9
        outch giOut[ichnl1], asig1, giOut[ichnl2],asig2, giOut[ichnl3],
↪ asig3, giOut[ichnl4],asig4
        elseif (imov == 2) then
            asig1, asig2 pan2 asig, 0.5
            ichnl1 random 1, 8
            ichnl1 = int(ichnl1)
            ichnl2 = ichnl1 + 1
            ichnl2 = ichnl2 % 9
            outch giOut[ichnl1], asig1, giOut[ichnl2],asig2
        endif
    endin
;; i 3 0 1 9 1 1
;; schedule 3, 0, 13, 0, 1, 4000, 300, 0.2, 1, 6
;; schedule 3.2, 0, 10, 0, 1, 2000, 100, 0.2, 3, 8
;; schedule 3.3, 0, 12, 0, 1, 800, 100, 0.2, 1, 4

;; schedule "stop", 0, 1, 3, 0

;; instr GlobalOut

;;     indx = 1
;;     until indx == 8 do
;;         gaMix[indx] limit gaMix[indx], -1, 0.8
;;         indx = indx + 1
;;     enduntil

;;     outch giOut[1], gaMix[1], giOut[2],gaMix[2], giOut[3], gaMix[3],
↪ giOut[4],gaMix[4], giOut[5], gaMix[5], giOut[6],gaMix[6], giOut[7],
↪ gaMix[7], giOut[8],gaMix[8]

;;     indx = 1
;;     until indx == 8 do
;;         gaMix[indx] = 0
;;         indx = indx + 1
;;     enduntil

;; endin

instr 30 ;; pattern generator for instr 3
;; for 3003
;;     kdurloop expon 8, p3, 0.2
;;     ilevel = 1
;;     klevel expon 1*ilevel, p3, 0.7*ilevel
;;     printk 1, kdurloop
;; end for 3003
loop:
;; for 3003

```

```

;; ilev = i(klevel)
;; idurloop = i(kdurloop)
;; print idurloop
;; end for 3003
ilev = 1
idurloop1 init p4
idurloop2 init p5
idurloop random idurloop1, idurloop2
ifndx random 0, 14 ; 0 - 14
ifndx = int(ifndx)
itimescal1 init p6
itimescal2 init p7
itimescal random itimescal1, itimescal2
ipitch1 init p8
ipitch2 init p9
ipitch random ipitch1, ipitch2
icf init p10
ibw init p11
imov init p12 ; 0 "cross", 1 "linear", 2 "static"
timeout 0, idurloop, play
reinit loop
play:
;; i 3 0 1 3 1 1 1000 50
event_i "i", 3, 0, 1, ifndx, itimescal, ipitch, icf, ibw, imov, ilev
endin

instr 3001 ;; Section generator for patgen 30
;; instr for testing permutations
idur = 30
idurloop1 = 1
idurloop2 = 10
itimescal1 = 0.4
itimescal2 = 2
ipitch1 = 0.4
ipitch2 = 2
icf = 0 ;; if 0 then no filter
ibw = 0
imov = 0 ; 0 "cross", 1 "linear", 2 "static"
schedule 30, 0, idur, idurloop1, idurloop2, itimescal1, itimescal2,
↪ ipitch1, ipitch2, icf, ibw, imov
endin

instr 3002 ;; Section generator for patgen 30
;; high pitched
idur = 60*3
idurloop1 = 50
idurloop2 = 120
itimescal1 = 4
itimescal2 = 5
ipitch1 = 5
ipitch2 = 7

```

```

    icf = 3000 ;; if 0 then no filter
    ibw = 300
    imov = 1 ; 0 "cross", 1 "linear", 2 "static"
    schedule 30, 0, idur, idurloop1, idurloop2, itimescal1, itimescal2,
    ↪ ipitch1, ipitch2, icf, ibw, imov
    endin

```

```

instr 3003 ;; Section generator for patgen 30
;; high density no processing cross
    idur = 60*5
    idurloop1 = 1
    idurloop2 = 3
    itimescal1 = 0.85
    itimescal2 = 1.2
    ipitch1 = 0.8
    ipitch2 = 1.2
    icf = 0 ;; if 0 then no filter
    ibw = 0
    imov = 0 ; 0 "cross", 1 "linear", 2 "static"
    schedule 30, 0, idur, idurloop1, idurloop2, itimescal1, itimescal2,
    ↪ ipitch1, ipitch2, icf, ibw, imov
    endin

```

```

instr 3004 ;; Section generator for patgen 30
;; low pitch filter static
    idur = 60*2
    idurloop1 = 5
    idurloop2 = 12
    itimescal1 = 0.3
    itimescal2 = 0.4
    ipitch1 = 0.2
    ipitch2 = 0.5
    icf = 200 ;; if 0 then no filter
    ibw = 200
    imov = 2 ; 0 "cross", 1 "linear", 2 "static"
    schedule 30, 0, idur, idurloop1, idurloop2, itimescal1, itimescal2,
    ↪ ipitch1, ipitch2, icf, ibw, imov
    endin

```

```

instr Test
    ;; kitem init 1
    ;; item = i(kitem)
    ;; print item
    ;; i1 = 8 % 9
    ;; print i1
    ;; i2 = 9 % 9
    ;; print i2
    ifn = giHeli0
    ilen = ftlen(ifn)/sr
    ktimescal init p4
    itimescal = i(ktimescal)

```

```

    print itimescal
    print ilen
endin

</CsInstruments>
<CsScore>
f1 0 8192 10 1
f2 0 8193 9 .25 1 0

;; i"GlobalOut" 0 3600
;; i "Test" 0 1 2

;i 3001 0 1
i 3002 0 1

e

```

```

</CsScore>
</CsoundSynthesizer>

```

A.3 Gendy Cloud Csound Code

```

    <CsoundSynthesizer>
    <CsOptions>
    -odac --m-amps=0
    </CsOptions>
    <CsInstruments>
    ;; GENDY CLOUD
    ;; Serkan Sevilgen
    ;; https://serkansevilgen.com
    ;; Sonified Symposium, ITU/MIAM

    sr = 48000
    ksmpts = 100
    Odbfs = 1
    nchnls = 20

    ;; seed for stochastic calculations
    seed 264736

    ;; Total Number of Active Gendy Instruments
    gitotal_num_perf = 16

    ;; OSC listening incoming messages address and port
    gSaddress = "/sonified22"
    giPort = 7770

    giOutSetup = 8 ;; Options, 1: binaural, 2: stereo, 8: first ring only,
    ↪ 12: two rings ;; NOT IMPLEMENTED YET

```

```

;; Ring8
giOutChn_1 = 1
giOutChn_2 = 2
giOutChn_3 = 4
giOutChn_4 = 6
giOutChn_5 = 8
giOutChn_6 = 7
giOutChn_7 = 5
giOutChn_8 = 3

;; Ring4
;; giOutChn_9 = 19
;; giOutChn_10 = 20
;; giOutChn_11 = 21
;; giOutChn_12 = 22

;;;; STEREO
;; giOutChn_1 = 1
;; giOutChn_2 = 2
;; giOutChn_3 = 1
;; giOutChn_4 = 2
;; giOutChn_5 = 1
;; giOutChn_6 = 2
;; giOutChn_7 = 1
;; giOutChn_8 = 2

giOut[] init 13
;; Out array starts with 1 instead of 0 for convenience during
↪ channel/speaker matching. first item in Out array is for the first
↪ speaker, etc.
giOut[] fillarray 0, giOutChn_1, giOutChn_2, giOutChn_3, giOutChn_4,
↪ giOutChn_5, giOutChn_6, giOutChn_7, giOutChn_8

giOutStereo[] init 2
giOutStereo[] fillarray 1, 2

;; gSparamtypes = "isf" ;; userid, param_name, param_val
giOscHandle OSCinit giPort

;; TOTAL NUMBER OF PERFORMERS
;; gitotal_num_perf = 2
gitotal_perf_params = 4
;; ARRAY FOR PERFORMANCE PARAMETERS (OSC)
gkPerfArr[][] init gitotal_num_perf, gitotal_perf_params
;; TODO: How to assign defaults programmatically below (with UDO?)
;; gkPerfArr[][] fillarray 1, 1, 1, 1, 1, 1, 1, 1
;; arr[userid] = [gate, note_dur, rest_dur, trig_gendy]

;; Additional Stereo Out
gaAudio2[] init 2

```

```

;; Global stereo reverb
gaReverbL init 0
gaReverbR init 0

;;;;;;;;;;;;;;;;;;;;;;;;;;
;; UDOS
;;;;;;;;;;;;;;;;;;;;;;;;;;

opcode setDefaults, 0, 0 ;; sets sane defaults to performance array
;; Default values in array is 0 when it is first created
;; note_dur can not be 0
;; gate should be 0
;; trig_gendy 1
;; gate, note_dur_indx, rest_dur_indx, trig_gendy
kRow[] init 4
kRow[] fillarray 0, 10, 10, 1
kuserid init 0 ;; also userid
while kuserid < gitotal_num_perf do
gkPerfArr setrow kRow, kuserid
kuserid += 1
od
endop

opcode getPerfIndx, k, S ;; returns index of a named parameter in perf
↪ array
Sname xin
kindx init 0
kgate strcmpk "gate", Sname
knote_dur strcmpk "note_dur", Sname
krest_dur strcmpk "rest_dur", Sname
ktrig_gendy strcmpk "trig_gendy", Sname

if kgate == 0 then
kindx = 0
elseif knote_dur == 0 then
kindx = 1
elseif krest_dur == 0 then
kindx = 2
elseif ktrig_gendy == 0 then
kindx = 3
endif
xout kindx
endop

opcode getPerfVal, k, kS ;; shortcut to get value from gkPerfArr, ins:
↪ userid and param index
kuserid, Sname xin
kindx getPerfIndx Sname
kRow[] getrow gkPerfArr, kuserid
xout kRow[kindx]

```

```

endop

opcode setPerfVal, 0, kSk
    kuserid, Sname, kval xin
    kindx getPerfIndx Sname
    kRow[] getrow gkPerfArr, kuserid
    kRow[kindx] = kval
;; printarray kRow
    gkPerfArr setrow kRow, kuserid
endop

opcode noteDurArr, k[], 0
    kArr[] fillarray 0.01,0.11,0.21,0.31,0.41,0.51,0.61,0.71,0.81,
↪ 0.91,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,6.,7.,8.,9.,10.,11.,12.,13.,14.,15.,16.,17.,18
xout kArr
endop

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; TEST UDOS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; TODO: DEFINE MOCK GLOBALS FOR ARRAYS AND VALUES. DO NOT WORK ON PROD
↪ VARIABLES

;; OPCODE TESTING HELPERS
gSsuccess = "SUCCESS"
gSfail = "FAIL"

opcode test_getPerfIndx, S, 0
    kval getPerfIndx "note_dur"
    Sout init ""
    if kval == 1 then
        Sout = gSsuccess
    else
        Sout = gSfail;
    endif
    Sout strcat "test_getPerfIndx: " , Sout
    xout Sout
endop

opcode test_setPerfVal, S, 0
setPerfVal 1, "note_dur", 3
kArr[] getrow gkPerfArr, 1
kval = kArr[1]
    if kval == 3 then
        Sout = gSsuccess
    else
        Sout = gSfail;
    endif
    Sout strcat "test_setPerfVal: " , Sout
    xout Sout

```

```

endop

instr TestSuite
    Sresult1 test_getPerfIndx
    prints Sresult1
    Sresult2 test_setPerfVal
    prints Sresult2
endin
;schedule "TestSuite", 0, 1

;;;;;;;;;;;;;
;; INIT INSTR
;;;;;;;;;;;;;

instr initAll
setDefaults
endin

schedule "initAll", 0, 1

;;;;;;;;;;;;;
;; ALWAYS ON INSTR
;;;;;;;;;;;;;

instr MessageRec ;; Listens incoming OSC msg. Always ON
    kuserid init 0
    Sname init ""
    Susername init ""
    kval init 0
    kans OSClisten giOscHandle, gSaddress, "isfs", kuserid, Sname, kval,
↪ Susername
    if kans == 1 then
        ;; kindx getPerfIndx Sname
        kindx = 99
        printf "userid: %i name: '%s' val: %f index: %i" , 1, kuserid,
↪ Sname, kval, kindx
        kgate strcmpk "gate", Sname
        printk 1, kgate
        setPerfVal kuserid, Sname, kval
        printarray gkPerfArr
    endif
endin

schedule "MessageRec", 0, -1
; schedule "stop", 0, 1, "MessageRec", 8

; schedule "msend", 0, 1, 1, "rest_dur", 2.4

instr Reverb
    kRoomSize init 0.5
    kHFDamp init 0.35

```

```

    aL, aR freeverb gaReverbL, gaReverbR, kRoomSize, kHFDamp, sr, 0
    outs aL, aR
    clear gaReverbL
    clear gaReverbR
endin

schedule "Reverb", 0, -1

instr outStereo
    outch giOutStereo[0], gaAudio2[0], giOutStereo[1], gaAudio2[1]
    gaAudio2[0] = 0
    gaAudio2[1] = 0
endin

schedule "outStereo", 0, -1

instr 99 ;; initiates patgen for each user
    print gitotal_num_perf
    iuserid init 0 ;; also userid
    while iuserid < gitotal_num_perf do
        iinstr_num = 8000 + (iuserid/100)
        ;; print iinstr_num
        schedule iinstr_num, 0, -1, iuserid
        ;; schedule "PatGen", 0, -1, icount
        iuserid += 1
    od
endin

schedule 99, 0, 1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; HELPER INSTRUMENTS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

instr stop
    Sname = p4
    imode = p5
    irel = 1
    insno nstrnum Sname
    ;; imode
    ;; turn off all instances (0), oldest only (1), or newest only (2)
    ;; 4: only turn off notes with exactly matching (fractional)
    ↪ instrument number
    ;; 8: only turn off notes with indefinite duration (p3 < 0 or MIDI)
    turnoff2 insno, imode, irel
endin

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

```

;; INSTRUMENTS
;;;;;;;;;;;;;;;;;;;;;;;;;;

instr 8 ;; Xenakis gendy
  kamp = p4 ;; 0.2 - 0.7
  kampdist = p5 ;; 1 - 6
  kdurdist = p6 ;; 0.0001 - 1
  kadpar = p7 ;; 0.0001 - 1
  kddpar = p8 ;; 0.0001 - 1
  kminfreq = p9 ;; 20 - 20000
  kmaxfreq = p10 ;; 20 - 20000
  kampscl = p11 ;; 0.1 - 1
  kdurscl = p12 ;; 0.1 - 1
  ideg = p13
  iuserid = p14
  kUserArr[] getrow gkPerfArr, iuserid
  kgate = kUserArr[0]

  asig gendy kamp, kampdist, kdurdist, kadpar, kddpar, kminfreq,
  ↪ kmaxfreq, kampscl, kdurscl

  ;; kenv1 linsegr 0, 0.001, 1, p3-0.002, 1, 0.001, 0
  ;; kenv2 linsegr 0, 0.05, 1, p3-0.1, 1, 0.05, 0
  kenv linsegr 0, 0.01, 1, p3-0.02, 1, 0.01, 0
  ;; if p3 < 0.1 then
  ;;   kenv = kenv1
  ;;   else
  ;;   kenv = kenv2
  ;; endif

  asig alpass asig, 1, 0.1
  ilevel random 0.2, 0.8
  asig = asig * kgate
  asig = asig*kenv*ilevel

ideg1 random 0, 360
ideg2 random -360, 360

if p3 < 1 then
  kalpha = ideg
else
  kalpha line ideg1, p3, ideg2
endif

kbeta = 0

koutArray init 8
;; irandomsp random 0, 3
;; if irandomsp < 2 then
;;   koutArray = 8
;; else

```

```

;; koutArray = 4
;; endif

if koutArray == 8 then
    ;create array for B-format 2nd order (9 chnls)
    aBform8[] init 9
    ; generate B-format
    aBform8 bformenc1 asig, kalpha, kbeta
    ;write out b-format
    ;; fout "fout.wav", 18, aBform
    ;create array for audio output (8 channels)
    aAudio8[] init 8
    ;decode B format for 8 channel circle loudspeaker setup
    aAudio8 bformdec2 6, aBform8

    ;; aBform2[] init 4
    ;; ; generate B-format
    ;; aBform2 bformenc1 asig, kalpha, kbeta
    ;; aAudio2[] init 2
    ;; aAudio2 bformdec2 1, aBform2
    ;; outch 19,aAudio2[0], 20, aAudio2[1]
    outch giOut[1], aAudio8[0],giOut[2], aAudio8[1],giOut[3],
    ↪ aAudio8[2],giOut[4], aAudio8[3],giOut[5], aAudio8[4], giOut[6],
    ↪ aAudio8[5],giOut[7], aAudio8[6],giOut[8], aAudio8[7]

;; else
;; ;create array for B-format 2nd order (4 chnls)
;; aBform4[] init 4
;; ; generate B-format
;; aBform4 bformenc1 asig, kalpha, kbeta
;; ;write out b-format
;; ;; fout "fout.wav", 18, aBform
;; ;create array for audio output (8 channels)
;; aAudio4[] init 4
;; ;decode B format for 4 channel circle loudspeaker setup
;; gaAudio4 bformdec2 2, aBform4

;; outch giOut[9], aAudio4[0],giOut[10], aAudio4[1],giOut[11],
    ↪ aAudio4[2],giOut[12], aAudio4[3]
endif
;; aL, aR pan2 asig, ideg
;; outs aL, aR
;; send to reverb ?? with ambisonics
;; iRvbAmt random 0.1, 0.8
;; gaReverbL = gaReverbL + (aL * iRvbAmt)
;; gaReverbR = gaReverbR + (aR * iRvbAmt)
endin

instr 8000 ;; Pattern generator for instr 8
kuserid = p4
loop:

```

```

ktrig getPerfVal kuserid, "trig_gendy"
kalert = 5
if ktrig == 1 then
    kampdist random 1, 6
    kamp random 0.05, 0.1
    kdurdist random 0.0001, 1
    kadpar random 0.0001, 1
    kddpar random 0.0001, 1
    kminfreq random 20, 2000
    kfreqdiff random 10, 1000
    kmaxfreq = kminfreq + kfreqdiff
    kampscl random 0.1, 1
    kdurscl random 0.1, 1
    setPerfVal kuserid, "trig_gendy", 0
endif

kdeg random 0, 360
;; TODO use getPerfVal
kUserArr[] getrow gkPerfArr, kuserid
knottedur_lambda = kUserArr[1]
krestdur_lambda = kUserArr[2]

kDurArr[] noteDurArr
ilenarr lenarray kDurArr
;; print ilenarr
knottedurIndx poisson knottedur_lambda
if knottedurIndx > ilenarr-1 then
    knottedurIndx = ilenarr-1
endif
knottedur = kDurArr[knottedurIndx]

krestdurIndx poisson krestdur_lambda
if krestdurIndx > ilenarr-1 then
    krestdurIndx = ilenarr-1
endif
krestdur = kDurArr[krestdurIndx]

kdurloop = knottedur+krestdur
idurloop = i(kdurloop)

timeout 0, idurloop, play
reinit loop
play:
    inotedur = i(knottedur)
    irestdur = i(krestdur)
    ;; printf "note: %f, rest: %f\n", 1, inotedur, irestdur
    iamp = i(kamp)
    iampdist = i(kampdist)
    idurdist = i(kdurdist)
    iadpar = i(kadpar)
    iddpar = i(kddpar)

```

```

    iminfreq = i(kminfreq)
    imaxfreq = i(kmaxfreq)
    iampscl = i(kampscl)
    idurscl = i(kdurscl)
    ideg = i(kdeg)
    iuserid = i(kuserid)
    inum = 8+((iuserid+1)/100)
    ;; print inum
    event_i "i", inum, 0, inotedur, iamp, iampdist, idurdist,
↪ iadpar, iddpar, iminfreq, imaxfreq, iampscl, idurscl, ideg, iuserid
endin

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;
;; TEMP INSTRUMENTS TO CHECK
;; STUFF DURING DEVELOPMENT
;;;;;;;;;;;;;;;;;;;;;;;;;

```

```

instr msend
    iport = giPort
    ;; "/xenakis-net/user1-HZHa7DqBQwyn", "f", kdur1
    iuserid = p4
    Sparam = p5;; 0: "gate", 1: "note_dur", 2: "rest_dur", 3:
↪ "trig_timbre" [0, 1]
    ival = p6

    OSCsend 1, "localhost", iport, gSaddress, "isf", iuserid, Sparam, ival
endin

```

```

instr concArr
    ;; iArr1[] fillarray 1,1
    ;; iArr2[] fillarray 2,2
    ;; iArr1[] genarray 1, 2, 3
    ;; iArr2[] genarray 4, 5
    ;; iArrFinal[] interleave iArr1, iArr2
    ;; iSorted[] sorta iArrFinal
    ;; printarray iSorted
    iArr[] fillarray 0.01,0.11,0.21,0.31,0.41,0.51,0.61,0.71,0.81,
↪ 0.91,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,6.,7.,8.,9.,10.,11.,12.,13.,14.,15.,16.,17.,18
    ilen lenarray iArr
    print ilen
    printarray iArr
endin

```

```

schedule "concArr", 0, 1

```

```

instr print_arr
    iPerfArr[] = gkPerfArr
    printarray iPerfArr
endin

```

```

instr testSpeakers

```

```

ich = p4
  aSin poscil 0dbfs/4, 440
  outch giOut[ich], aSin*0.2
endin

instr runTestSpeaker
  schedule "testSpeakers", 3, 3, 1
  schedule "testSpeakers", 6, 3, 2
  schedule "testSpeakers", 9, 3, 3
  schedule "testSpeakers", 12, 3, 4
  schedule "testSpeakers", 15, 3, 5
  schedule "testSpeakers", 18, 3, 6
  schedule "testSpeakers", 21, 3, 7
  schedule "testSpeakers", 3, 3, 8
  schedule "testSpeakers", 6, 3, 9
  schedule "testSpeakers", 9, 3, 10
  schedule "testSpeakers", 12, 3, 11
  schedule "testSpeakers", 15, 3, 12
endin

;; schedule "print_arr", 0, 1

</CsInstruments>
<CsScore>
; i "MidiInput" 0 999
;; i "CheckArr" 0 999
;; i "OSCListener" 0 300
;; i "Run" 0 10
;; i 8000 0 1 0
;; i 2 0 5 5
;; i "msend" 0 1
;; i "mrec" 0 1
;; i 99 0 3600
;; i "runTestSpeaker" 0 24
;; i "testSpeakers" 0 60
f 0 3600
</CsScore>
</CsSoundSynthesizer>

```