Pattern modelling and pattern processing in image and speech signals

Estudis: Enginyeria de Telecomunicació

Autor: Narcís Sayols Baixeras

Directors: Francesc Vallverdú Bayes
           Sebastià Xambó Descamps

Any: 2011-2012
Final Project at ETSETB

Pattern modelling and pattern processing in image and speech signals

Narcís Sayols Baixeras

Advisor: Francesc Vallverdú Bayés
Coadvisor: Sebastià Xambó Descamps

Departament de Teoria del Senyal i Comunicacions
Abstract

Key words: Information And Communication, Circuits Communication, information, Pattern theory, Wavelet analysis

In the context of signal processing [31, 43, 34, 35], the aim of this memoir is to explore the pattern theory (PT) of Grenander and Mumford ([21, 38]) by focusing on two proof-of-concept applications, one dealing with images and the other with voice. The image application not only classifies the symmetry type of mosaics given a digital picture, but also can generate mosaic images with virtually any pattern decoration. The voice application classifies the phonemes in a digital voice recording and also can generate virtually unlimited variations on its pronunciation.

These applications illustrate, in ways that are described in detail in the memoir, some of the key advantages of PT over and beyond the more familiar approaches known as pattern recognition and pattern classification ([11, 58, 2]). One important advantage is a sort of ‘division of labor’: the construction of pattern models, which takes into account the data and is eminently theoretical, is separate from the pattern processing work (implementation, algorithms, programs). A second advantage is that pattern analysis (recognition, classification) and pattern synthesis (generating random instances that ‘look and feel’ just as the world signals being modeled) become two sides of the same coin, in the sense that the same approach provides both. Actually, analysis is done via synthesis, because the inspiration for the proposed models comes from the data and recognition is resolved by using suitable plausibility rules in order to find the best pattern parameters fitting the data. Another advantage of the approach is its resilience in front of noisy and ambiguous samples. Finally, it is important to remark that pattern models can often be reused on data that are totally unrelated to the data that led to those models.

The material in this memoir has been grouped into three parts. The first part, consisting of the first four chapters, is best described as a presentation of the theoretical background: Wavelet analysis (Ch. 1), Mosaic groups (Ch. 2), Phonetic techniques (Ch. 3) and Pattern Theory (Ch. 4). These chapters can be read independently, except that Ch. 4 occasionally refers to the preceding materials. The second part, consisting of two chapters, is focussed on the two applications referred to above: Wallpaper group generation and classification (Ch. 5) and Phonetic recognition and pronunciation (Ch. 6). Chapter 5 depends on chapters 1 and 4, while Chapter 6 depends on chapters 3 and 4. For reference convenience, we include an appendix that summarizes the signal processing ideas and results that we use throughout.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td><strong>Chapter 1. Wavelet Theory</strong></td>
<td>3</td>
</tr>
<tr>
<td>1.1. Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.2. Wavelet matrices</td>
<td>4</td>
</tr>
<tr>
<td>1.3. Wavelet systems</td>
<td>6</td>
</tr>
<tr>
<td>1.4. Biorthogonal wavelet systems</td>
<td>8</td>
</tr>
<tr>
<td>1.5. Multiresolutions and the Mallat algorithm</td>
<td>9</td>
</tr>
<tr>
<td>1.6. The Mallat algorithm for periodic data</td>
<td>12</td>
</tr>
<tr>
<td>1.7. Wavelet image compression</td>
<td>13</td>
</tr>
<tr>
<td>1.8. Wavelet channel coding</td>
<td>14</td>
</tr>
<tr>
<td><strong>Chapter 2. Wallpaper groups</strong></td>
<td>19</td>
</tr>
<tr>
<td>2.1. Euclidean plane isometries</td>
<td>19</td>
</tr>
<tr>
<td>2.2. Wallpaper groups</td>
<td>21</td>
</tr>
<tr>
<td>2.3. Wallpaper group notations</td>
<td>22</td>
</tr>
<tr>
<td>2.4. The 17 wallpaper groups</td>
<td>23</td>
</tr>
<tr>
<td><strong>Chapter 3. Phonetic techniques</strong></td>
<td>27</td>
</tr>
<tr>
<td>3.1. Introduction</td>
<td>27</td>
</tr>
<tr>
<td>3.2. Phonetic sounds</td>
<td>28</td>
</tr>
<tr>
<td>3.3. Markov models</td>
<td>29</td>
</tr>
<tr>
<td>3.4. Speech recognition</td>
<td>30</td>
</tr>
<tr>
<td>3.5. N-gram approximation</td>
<td>32</td>
</tr>
<tr>
<td>3.6. Feature extraction</td>
<td>33</td>
</tr>
<tr>
<td>3.7. Text Normalisation</td>
<td>34</td>
</tr>
<tr>
<td>3.8. Phonetic analysis</td>
<td>34</td>
</tr>
<tr>
<td>3.9. Prosodic analysis</td>
<td>35</td>
</tr>
<tr>
<td>3.10. Waveform synthesis</td>
<td>36</td>
</tr>
<tr>
<td><strong>Chapter 4. Pattern Theory</strong></td>
<td>37</td>
</tr>
<tr>
<td>4.1. Introduction</td>
<td>37</td>
</tr>
<tr>
<td>4.2. Pattern formalism</td>
<td>38</td>
</tr>
<tr>
<td>4.3. Bayesian probability</td>
<td>39</td>
</tr>
<tr>
<td>4.4. Pattern analysis and Pattern synthesis</td>
<td>41</td>
</tr>
</tbody>
</table>
4.5. Machine translation model

Chapter 5. Wallpaper group algorithm
5.1. Mosaic generator
5.2. Decoration generator
5.3. Main cell creation
5.4. Tiling
5.5. Classification algorithm
5.6. Translational lattice detection
5.7. Classifying the wallpaper groups
5.8. Examples of the algorithm
5.9. Classification of generated mosaics
5.10. MATLAB functions

Chapter 6. Speech classification
6.1. Classification algorithm
6.2. Speech segmentation
6.3. Phoneme recognition
6.4. Examples of the classification algorithm
6.5. Synthesis algorithm
6.6. Phoneme generator
6.7. MATLAB functions

Conclusions and future work

Appendix A. Neural Networks
Appendix B. Mosaic's Code
Appendix C. Speech Code
References
Introduction

Signal processing is an area in engineering and applied mathematics that deals with operations on or analysis of signals to extract or modify information. Signals is a loosely defined term that includes a variety of forms, such as images, sounds, sensor data, control system signals, or telecommunication transmission systems (radio or television signals, for example). The importance of signal processing stems from the fact that it provides theoretical and practical tools to restore signals distorted by noise and, more generally, to cull information that is normally hidden in the raw data.

In this context, the main focus of the present project is to probe the potential of pattern theory, an insightful framework which is capable, we believe, of shedding new light on old problems and of steering novel developments, theoretical and practical, in the years to come.

Pattern theory tries to analyze all types of signals (images, sounds, text, protein chains, etc.), using statistical models, mostly Bayesian, that describe them. Pattern theory in this sense is not to be confused with mere ‘pattern recognition’ or ‘pattern classification’, two important classic areas of signal processing which, in the approach we adopt, are dovetailed as pattern processing analysis and synthesis. In other words, while in those areas the analysis and synthesis of signals are two different problems, in pattern theory they are two aspects of the same model.

A bonus aspect to be stressed here is that pattern theory models used in certain types of signals can often be reused in other areas. The root of this is that pattern theory facilitates the recognition of similar structures in different problems, so that useful models for a specific set of features in a signal may also be suitable for modeling features in other apparently unrelated signals.

As a prerequisite basis for pattern theory, this project includes also a presentation of wavelet analysis. Wavelet theory amounts to a sweeping generalization of Fourier analysis that allows to ‘solve’ local signals in both time and frequency. In recent years, this technique has become an indispensable tool for treating signals whose relevant characteristics are localized. In our study we consider the mathematical fundamentals together with examples of their use, for instance in compressing and filtering of signals.

We view wavelet theory as a crucial ingredient for our future investigations in pattern theory and, more specifically, in image grammars, which are supposed to provide a suitable support for the description of images as a hierarchical structure of parts and relations among them. We also envision the use pattern and wavelet theory in medical imaging. In this vast area there is an increasing need for fast and reliable processing that can help clinical and radiology researchers and practitioners. So, we will try to model the
symptoms of disease lesions as they appear in the images, as for example lesions of multiple sclerosis in nuclear magnetic resonance.

The present memoir has two main parts. The first part, whose content is mainly theoretical, is devoted to wavelet analysis and pattern theory. Background material that is needed for the applications in the second part is also included here: wallpaper groups and basic phonetic techniques (voice synthesis and recognition methods).

The second part of the project deals with two applications, one in image processing and the other in voice processing. In both cases, the application includes a detailed description of the algorithms and of their implementations in MATLAB. The image processing application illustrates pattern analysis, in the sense that it finds the symmetry group of a wallpaper image, and pattern synthesis, as it generates wallpaper patterns in any of the 17 possible symmetry groups. Similarly, the voice processing application illustrates pattern analysis and synthesis by producing the phoneme transcription of a speech signal and voice renderings of phoneme sequences.

The memoir has been organized as follows:

The aim of Chapter 1 is the study of wavelet functions, their proprieties and their uses to compress signals. The wavelet transforms, as tools for signal and system analysis, are also described.

Chapter 2 presents the theory of wallpaper groups needed in Chapter 5. We describe all possible wallpaper groups (also called plane crystallographic groups) and we summarize the properties that single out one group from the others.

Chapter 3 is devoted to the phonetic techniques needed in Chapter 6. We also present different algorithms for speech recognition and speech synthesis.

In Chapter 4, we introduce a mathematical approach to pattern theory. We present the needed Bayesian theory and show that in pattern theory the analysis and synthesis are two aspects of the same problem.

In the last two chapters we explore some applications of the preceding theories. The problems we consider, which belong to the image processing domain (Chapter 5) and the speech processing domain (Chapter 6), are in particular meant to illustrate the remarkable fact that the same theoretical ideas and results can be applied in both domains. In more detail, in Chapter 5 we present an algorithm, and an implementation in MATLAB, that returns, given an image of a wallpaper pattern, the wallpaper group of that pattern. The chapter ends with some examples to illustrate how the algorithm and the implementation work. Finally, in Chapter 6 we present an application (algorithm and implementation in MATLAB) that parses the phonemes of a given recorded voice.

The appendixes provide an introduction to artificial neural networks (Appendix A) and the MATLAB code of the algorithms (Appendix B and Appendix C).
Chapter 1

Wavelet Theory

In this chapter we summarize the basic notions and results from wavelet theory that are needed in Chapter 6 and, to a lesser extent, in Chapter 5. The main general references we have used are [45, 14, 35, 54, 61, 9].

1.1. Introduction

To assess the relevance of wavelet theory, we start by considering some strengths and weaknesses of Fourier analysis. The main strength is Fourier’s theorem. This epoch-making result applies to periodic complex functions \( f(x) \) of a real variable \( x \) which is (Lebesgue) integrable in a period interval. If \( f(x) \) has period \( T \) and \( \omega = 2\pi/T \) (fundamental angular frequency), the theorem asserts that it can be represented as the summation (Fourier series) of simple harmonic vibrations with angular frequencies \( \omega_n = n\omega \) \( (n \geq 0) \). These vibrations are dilations and translations of the simple wave \( \sin(x) \) and the approximations of \( f(x) \) given by the partial sums of the Fourier series are optimal with respect to ‘energy’ (the \( L^2 \) norm). These reasons explain the success of Fourier analysis ever since its discovery and why it will continue to play an indispensable role in the future.

The main weaknesses of Fourier analysis with regard to signal processing are that it cannot characterize signals locally in the time domain and that is not very adequate for non stationary signals. Consequently, Fourier approximations are not very suitable to represent images and patterns that contain transient or localized components. In the case of pattern recognition, for example, many important features are highly localized in spatial regions. Wavelet theory, which amounts to a far-reaching generalization of the Fourier techniques, was intentionally designed to overcome these defects by focussing on ‘wavelet signals’, which are localized in both time and frequency.

For historical remarks on the development of these ideas, [61, 8, 35, 54] are convenient references. In particular, wavelet analysis has been especially successful in pattern recognition: see [54] for a general approach and [13, 64] for the more specific case of speech recognition.

Although not explicitly used in this memoir, a closely related topic with strong recent developments is ‘sparse compression’. This has given new life to wavelet theory, as expressed by S. Mallat in [35]: “Wavelets are no longer the central topic, despite the previous edition’s original title. It is just an important tool, as the Fourier transform is. Sparse representation and processing are now at the core”. The problem solved by the sparse representation [5, 6, 41, 25, 29] is to search for the most compact representation of a signal in terms...
1. Wavelet theory

of a linear combination of some suitable basis. These ideas lead to ‘sparse processing’ techniques, which
use, among others, wavelet or curvelet transforms \cite{35,4} as tools to find optimal samplings of a signal.

1.2. Wavelet matrices

Henceforth in this chapter, in our theoretical approach we will follow \cite{45}.

**Definition.** Let \( \mathbb{F} \) be a subfield of the field \( \mathbb{C} \) of complex numbers. Consider an array \( A = (a^s_r) \) consisting of \( m \) rows of possibly infinite length,

\[
A = \begin{pmatrix}
... & a^0_{0} & a^0_{1} & a^0_{2} & ... \\
... & a^1_{0} & a^1_{1} & a^1_{2} & ... \\
... & a^{m-1}_{0} & a^{m-1}_{1} & a^{m-1}_{2} & ... \\
... & ... & ... & ... & ... \\
... & ... & ... & ... & ...
\end{pmatrix}
\]

where each \( a^s_r \in \mathbb{F} \) and \( m \geq 2 \). We will refer to this array \( A \) as a matrix, even though it may have infinitely many columns.

For \( l \in \mathbb{Z} \), define submatrices \( A_l \) of \( A \) of size \( m \times m \) as follows:

\( A_l = (a^s_{lm+r}), \quad r = 0, ..., m-1, \quad s = 0, ..., m-1. \)

These submatrices amount to a partition of \( A \) into \( m \times m \) blocks that can be expressed as

\( A = (\ldots, A_{-1}, A_{0}, A_{1}, A_{2}, \ldots). \)

**Definition.** The Laurent series of the matrix \( A \) is the formal power series

\( A(z) = \sum_{l=-\infty}^{+\infty} A_l z^l. \)

More explicitly,

\[
A(z) = \begin{pmatrix}
\sum_{k=-\infty}^{+\infty} a^0_{mk-l} z^k & \ldots & \sum_{k=-\infty}^{+\infty} a^0_{mk-1-l} z^k \\
\vdots & \ddots & \vdots \\
\sum_{k=-\infty}^{+\infty} a^{m-1}_{mk-l} z^k & \ldots & \sum_{k=-\infty}^{+\infty} a^{m-1}_{mk-1-l} z^k
\end{pmatrix}
\]

**Definition.** Assume that \( A \) has only a finite number of columns. Then we can write

\( A(z) = \sum_{l=N_1}^{N_2} A_l z^l, \)

where we assume that \( A_{N_1} \) and \( A_{N_2} \) are nonzero matrices. The number \( g = N_2 - N_1 \) is called the genus of the Laurent series \( A(z). \)

Although some of the definitions below make perfect sense when \( A \) has infinitely many non-zero columns, henceforth we will assume that \( N_1 = 0 \) and \( N_2 = mg - 1 \) (so \( g \) is the genus). In particular, all infinite sums below have only finitely many non-zero summands.
**Definition.** The *adjoint* of the Laurent matrix $A(z)$, $\tilde{A}(z)$, is

$$
\tilde{A}(z) = A^*(z^{-1}) = \sum_{l=-\infty}^{+\infty} A_l^* z^{-l},
$$

where $A_l^*$ is the conjugate transpose of $A_l$.

**Definition.** The matrix $A$ is said to be a *wavelet matrix of rank $m$* if

- $A(z)\tilde{A}(z) = mI_m$
- $\sum_{k=-\infty}^{+\infty} a_k^* = m\delta^{s,0}$, $0 \leq s \leq m - 1$ where $\delta^{s,j}$ is the Kronecker function.

In terms of the matrix $A$, the first condition can be written as

$$
\sum_{k=-\infty}^{+\infty} a_{k+ml}^* \bar{a}_{k+ml} = m\delta^{s',s}\delta_{p,j}.
$$

This condition asserts that the rows $a^* = (a^*_0, \ldots, a^*_m, a^-_{y-1})$ have length equal to $\sqrt{m}$ and that they are pairwise orthogonal when shifted by an arbitrary multiple of $m$.

The second condition of the definition distinguishes between the first row and the other ones. The first row will be called *scaling vector* and the others *wavelet vectors*. So, this condition simply says that the sum of the components of the scaling vector has value $m$ while the sum of the components of the other rows is zero.

**Proposition.** A wavelet matrix with $m$ rows has rank $m$ in the classical sense.

Henceforth, the set of wavelet matrices of rank $m$ and genus $g$, with coefficients in the field $\mathbb{F}$, will be denoted $\text{WM}(m, g; \mathbb{F})$.

**Examples.**

1. **Haar matrices.** The wavelet matrices of genus 1 are called *Haar wavelet matrices*. Notice that the condition for an $m \times m$ matrix $H$ to be a Haar matrix is that $HH^* = mI_m$. For instance, the matrices

$$
\begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}
\quad
\begin{pmatrix}
1 & 1 \\
-1 & 1
\end{pmatrix}
$$

are both wavelet matrices of rank 2 and, as one can check, they are the only square wavelet matrices of rank 2 with real coefficients. More generally if we allow complex coefficients, we can easily see that the general complex Haar wavelet matrix of rank 2 has the form for $\theta \in \mathbb{R}$

$$
\begin{pmatrix}
1 & 1 \\
-e^{i\theta} & e^{i\theta}
\end{pmatrix}
$$

The matrix $\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$ will be called canonical Haar matrix of rank 2.

2. **Daubechies Wavelet Matrix of rank 2 and genus 2.** The matrix

$$
D_2 = \frac{1}{4} \begin{pmatrix}
1 + \sqrt{3} & 3 + \sqrt{3} & 3 - \sqrt{3} & 1 - \sqrt{3} \\
-1 + \sqrt{3} & 3 - \sqrt{3} & -3 - \sqrt{3} & 1 + \sqrt{3}
\end{pmatrix},
$$

discovered by Daubechies [7], is a rank 2 wavelet matrix.
1. WAVELET THEORY

(3) The Discrete Fourier Transform Matrix: Let \( m > 1 \) be an integer and \( w = e^{2\pi i/m} \) be a primitive \( m \)-th root of unity. The Discrete Fourier Transform matrix (DFT) of rank \( m \) defined as

\[
W = \begin{pmatrix}
1 & 1 & 1 & 1 & \ldots & 1 \\
1 & w & w^2 & w^3 & \ldots & w^{N-1} \\
1 & w^2 & w^4 & w^6 & \ldots & w^{2(m-1)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & w^{m-1} & w^{2(m-1)} & w^{3(m-1)} & \ldots & w^{(m-1)(m-1)}
\end{pmatrix},
\]

is a Haar matrix over \( \mathbb{C} \). If \( N = 2 \), then \( W \) is defined over \( \mathbb{R} \), and it coincides with \[
\begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}.
\]

(4) Flat Wavelet Matrices: These are defined as the wavelet matrices all of whose entries have the same absolute value. The Haar matrices of the previous example have this property. We present and example of a flat wavelet matrix of rank 2 and genus 4

\[
\begin{pmatrix}
1 & 1 & 1 & -1 \\
1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1
\end{pmatrix}.
\]

1.3. Wavelet systems

Let \( A \in WM(m, g; \mathbb{C}) \) be a wavelet matrix and consider the functional difference equation

\[
\varphi(x) = \sum_{k=0}^{mg-1} a_k^0 \varphi(mx - k)
\]

**Definition.** This equation is called the scaling equation associated with the wavelet matrix \( A \). If \( \varphi \) is a solution of this equation, then \( \varphi \) is called a scaling function associated with \( A \).

**Definition.** We define the wavelet functions \( \{\psi^1, \ldots, \psi^{m-1}\} \) associated with the wavelet matrix \( A \) and the scaling function \( \varphi \) by the formula

\[
\psi^s(x) = \sum_{k=0}^{mg-1} a_k^s \varphi(mx - k)
\]

Notice that if we define \( \psi^0 \) using the same formula, then \( \psi^0 = \varphi \) by the scaling equation.

**Theorem 1.** Let \( A \in WM(m, g; \mathbb{C}) \) be a wavelet matrix. Then, there exists a unique \( \varphi \in L^2(\mathbb{R}) \) such that (see [45]):

1. \( \varphi \) satisfies the scaling equation, \( \varphi(x) = \sum_{k=0}^{mg-1} a_k^0 \varphi(mx - k) \) and
2. \( \int_{\mathbb{R}} \varphi(x) dx = 1 \) (normalization condition).

Furthermore, this \( \varphi \) satisfies that \( \text{Supp}(\varphi) \subset [0, (g - 1)(\frac{m-1}{m}) + 1] \).

Let \( A \) be a wavelet matrix of rank \( m \) and \( \{\varphi, \psi^1, \ldots, \psi^{m-1}\} \) its scaling and wavelet functions. For \( k, j \in \mathbb{Z} \), we define

\[
\varphi_{jk} = m^j \varphi(mx - k) \\
\psi_{jk}^s(x) = m^j \psi^s(mx - k)
\]

These are the rescaled and translated scaling and wavelet functions. We introduce the notation

\[
\varphi_k(x) = \varphi_{0k}(x)
\]
1.3. WAVELET SYSTEMS

**Definition.** The wavelet system $W[A]$ associated with the wavelet matrix $A$ is the collection of functions

$W[A] = \{\phi_k(x), k \in \mathbb{Z}\} \cup \{\psi_{jk}^s(x), j, k \in \mathbb{Z}, j \geq 0, s = 1,\ldots, m - 1\}$

**Theorem 2.** Let $A \in WM(m, g; \mathbb{C})$, let $W[A]$ its wavelet system and let $f \in L^2(\mathbb{R})$. Then, there exists an $L^2$-convergent expansion

$$f(x) = \sum_{k=-\infty}^{+\infty} c_k \phi_k(x) + \sum_{s=1}^{m-1} \sum_{j=0}^{+\infty} \sum_{k=-\infty}^{+\infty} d_{jk}^s \psi_{jk}^s(x)$$

where the coefficients are given by

$$c_k = \int_{\mathbb{R}} f(x) \phi_k(x) dx$$

$$d_{jk}^s = \int_{\mathbb{R}} f(x) \psi_{jk}^s(x) dx$$

**Definition.** Let $\mathcal{H}$ be a Hilbert space with an inner product $\langle \cdot, \cdot \rangle$, and let $\{h_\alpha\}$ a countable set in $\mathcal{H}$. Then $\{h_\alpha\}$ is a tight frame if, for some $c \geq 0$,

$$f = c \sum_{\alpha \in I} \langle h_\alpha, f \rangle h_\alpha$$

for each $f \in \mathcal{H}$.

**Observation.** An orthonormal basis for $\mathcal{H}$ has this property with $c = 1$.

**Observation.** Although the most common wavelet matrices have an orthonormal wavelet system, it is important to note that the preceding theorem still holds if the wavelet system of the matrix is a tight frame. An example of wavelet tight frame which is not orthonormal can be found in [50].

**Examples.**

1. **The Haar functions.** If $H$ is the canonical Haar wavelet matrix of rank 2, then the scaling and wavelet functions $\phi$ and $\psi$ given by Theorem 1 can be determined in an elementary fashion: Their graphs are shown in Figure 1.

   ![Fig. 1. Haar scaling function (red) and Haar wavelet function (green).](image)

2. **Daubechies Wavelet functions.** Let $D_2 \in WM(2, 2, \mathbb{R})$ be the rank 2 wavelet matrix introduced in the Examples in 1.2 (example 2), and let $\phi$ and $\psi$ be the corresponding scaling and wavelet functions (see Figure 2 where they are drawn in red and green, respectively); the common support of $\phi$ and the wavelet function $\psi$ is $[0, 3]$. 
1.4. Biorthogonal wavelet systems

**Definition.** Let $L = (a^j_i)$ and $R = (b^j_i)$ be $m \times mg$ matrices with complex entries. We will say that $(L, R)$ is a wavelet matrix pair of rang $m$ and genus $g$ if

\[ L(z) \tilde{R}(z) = mI_m \]

and

\[ \sum_{k=0}^{mg-1} a^r_k = \sum_{k=0}^{mg-1} b^r_k = m \delta_{r,0} \]

Let $(L, R)$ be a wavelet matrix pair and $\{\phi, \Phi, \psi_r, \Psi_r, r = 1, \ldots, m - 1\}$ functions satisfying the scaling and wavelet equations

\[
\phi(x) = \sum_{k=0}^{mg-1} a^0_k \phi(mx - k)
\]

\[
\psi^r = \sum_{k=0}^{mg-1} a^r_k \phi(mx - k)
\]

\[
\Phi(x) = \sum_{k=0}^{mg-1} b^0_k \Phi(mx - k)
\]

\[
\Psi^r = \sum_{k=0}^{mg-1} b^r_k \Phi(mx - k)
\]

for $r = 1, \ldots, m-1$. The functions $\{\phi, \Phi\}$ and $\{\psi^r, \Psi^r, r = 1, \ldots, m - 1\}$ are called biorthogonal scaling functions and biorthogonal wavelet functions, respectively.

**Definition.** A biorthogonal wavelet system, $W[L, R]$, associated with a wavelet matrix pair $(L, R)$ are the rescaling and translates of its biorthogonal scaling and wavelet functions

\[ W[L, R] = \{\phi_k(x), \Phi_k(x), k \in \mathbb{Z}\} \cup \{\psi^r_{jk}, \Psi^r_{jk}, j, k \in \mathbb{Z}, j \geq 0, r = 1, \ldots, m - 1\} \]
Observation. A biorthogonal wavelet system satisfies the following biorthogonality properties:

\[ \int \varphi_k \Phi_{k'} = \delta_{k,k'} \]
\[ \int \varphi_k \Psi_{jk'} = 0, \quad r = 1, \ldots, m - 1 \]
\[ \int \psi_{jk} \Phi_{k'} = 0, \quad r = 1, \ldots, m - 1 \]
\[ \int \psi_{jk} \Psi_{jk'} = \delta_{j,j'} \delta_{k,k'} \]

Theorem 3. If \((L, R)\) is a wavelet matrix pair, then there exists an associated biorthogonal wavelet system \([\varphi_k, \psi_{jk}, \Phi_k, \Psi_{jk}]\) and, moreover, for any \(f \in L^2(R)\),

\[ f(x) = \sum_{k=-\infty}^{+\infty} \langle f, \varphi_k \rangle \Phi_k(x) + \sum_{j=0}^{m-1} \sum_{k=-\infty}^{+\infty} \langle f, \psi_{jk} \rangle \Psi_{jk}(x) \]

where the convergence is in the weak \(L^2\) sense.

Example.

1. Wavelet matrices: Every wavelet matrix \(A\) defines a wavelet matrix pair \((A, A)\). An example could be the Haar matrix of rank 2

\[ H_L = H_R = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \]

2. Rank 2 and genus 2: Here is an example of a wavelet matrix pair of rank \(m = 2\) and genus \(g = 2\):

\[ L = \begin{pmatrix} 1/3 & 2/3 & 2/3 & 1/3 \\ -1 & -2 & 2 & 1 \end{pmatrix} \]
\[ R = \begin{pmatrix} -1 & 2 & 2 & -1 \\ 1/3 & -2/3 & 2/3 & -1/3 \end{pmatrix} \]

3. Daubechies type, genus 3: A complex wavelet matrix pair for \(m = 2, g = 3\).

\[ L = \begin{pmatrix} -0.13 + 0.07i & 0.11 + 0.21i & 1.02 + 0.14i & 1.02 - 0.14i & 0.11 - 0.21i & -0.13 - 0.07i \\ -0.13 - 0.07i & -0.11 + 0.21i & 1.02 - 0.14i & -1.02 - 0.14i & 0.11 + 0.21i & 0.13 - 0.07i \end{pmatrix} \]
\[ R = \begin{pmatrix} -0.13 - 0.07i & 0.11 - 0.21i & 1.02 - 0.14i & 1.02 + 0.14i & 0.11 + 0.21i & -0.13 + 0.07i \\ -0.13 + 0.07i & -0.11 - 0.21i & 1.02 + 0.14i & -1.02 + 0.14i & 0.11 - 0.21i & 0.13 + 0.07i \end{pmatrix} \]

1.5. Multiresolutions and the Mallat algorithm

Let

\[ A = \begin{pmatrix} a_0 & a_1 & \ldots & a_{2^g-1} \\ b_0 & b_1 & \ldots & b_{2^g-1} \end{pmatrix} \]

be a wavelet matrix of rank \(2\) and genus \(g\), and let \(W = \{\varphi_k, \psi_{jk}\}\) be the corresponding orthonormal wavelet system. Introduce

\[ \varphi_{jk} = 2^{j/2} \varphi(2^j x - k) \]
the rescaling and translations of the scaling function $\varphi(x)$. We assume orthonormality for convenience of exposition, although the analysis we present here is valid in general case of tight frames. For $j \in \mathbb{Z}$, define the vector space

$$V_j = \text{Span}(\varphi_{jk} \mid k \in \mathbb{Z})$$

It can be proved that $L^2(\mathbb{R}) = \bigcup_j V_j$, so we have the orthogonal projections $P_j : L^2(\mathbb{R}) \rightarrow V_j$. Hence, given a function $f \in L^2(\mathbb{R})$, $P_j f$ converges to $f$, as $j \rightarrow \infty$, in the $L^2$ norm and its coefficients are given by the formula

$$P_j f(x) = \sum_{k=-\infty}^{+\infty} c_{jk} \varphi_{jk}(x)$$

where

$$c_{jk} = \int_{\mathbb{R}} f(x) \varphi_{jk}(x) \, dx$$

since the $\varphi_{jk}$'s are a tight frame for $V_j$. If we let $W_j$ denote the orthogonal complement of $V_j$ in $V_{j+1}$, then for a fixed $J \in \mathbb{Z}^+$

$$V_0 \oplus W_0 \oplus \cdots \oplus W_{J-1} = V_J$$

Moreover, if

$$f(x) = \sum_{k=-\infty}^{+\infty} c_{0k} \varphi_{0k}(x) + \sum_{j=0}^{+\infty} \sum_{k=-\infty}^{+\infty} d_{jk} \psi_{jk}(x)$$

for $f \in L^2(\mathbb{R})$, then we obtain

$$\sum_{k=-\infty}^{+\infty} c_{jk} \varphi_{jk}(x) = \sum_{k=-\infty}^{+\infty} c_{0k} \varphi_{0k}(x) + \sum_{j=0}^{J-1} \sum_{k=-\infty}^{+\infty} d_{jk} \psi_{jk}(x)$$

So, one can determine the coefficients on the right-hand side of the previous equation in terms of the coefficients on the left-hand side, and conversely. This is the Mallat algorithm.

Here is how the Mallat algorithm works: Suppose we are given an expansion of the form

$$f' = \sum_{k=-\infty}^{+\infty} c_{jk} \varphi_{jk}(x),$$

and we want to determine the coefficients of the corresponding lower-order expansion of the type

$$f' = \sum_{k=-\infty}^{+\infty} c_{0k} \varphi_{0k}(x) + \sum_{j=0}^{J-1} \sum_{k=-\infty}^{+\infty} d_{jk} \psi_{jk}(x)$$

We will do this successively in stages. First, we consider the decomposition

$$V_{J-1} \oplus W_{J-1} = V_J,$$

and determine the expansion coefficients in $V_{J-1}$ and $W_{J-1}$ for the coefficients in $V_J$. Thus, we can write

$$\sum c_{J-1,k} \varphi_{J-1,k} + \sum d_{J-1,k} \psi_{J-1,k} = \sum c_{jk} \varphi_{jk}$$

1The span of $S$ is defined as the set of all linear combinations, not necessarily finite, of elements of $S$. 
and try to determine the pair \( \{c_{J-1,k}, d_{J-1,k}\} \) in terms of the \( \{c_Jk\} \), and conversely. First, multiply the last equation by \( \varphi_{J-1,j} \) and integrate. We obtain:

\[
c_{J-1,l} = \sum c_{jk} \int \varphi_{J,k}(x) \varphi_{J-1,l}(x) \, dx
\]

Now use the basic scaling equation and wavelet defining equation to write

\[
\varphi_{J-1,l}(x) = \frac{1}{\sqrt{2}} \sum_{r=0}^{2^J-1} a_r \varphi_{J+1,l}(x) \quad (*)
\]

\[
\psi_{J-1,l}(x) = \frac{1}{\sqrt{2}} \sum_{r=0}^{2^J-1} b_r \varphi_{J+1,l}(x) \quad (**)
\]

Substituting (*) into (**) and using the orthogonality, we obtain

\[
c_{J-1,l} = \frac{1}{\sqrt{2}} \sum c_{J,r} a_r - 2l.
\]

Similarly, we find that

\[
d_{J-1,l} = \frac{1}{\sqrt{2}} \sum c_{J,r} b_r - 2l.
\]

More generally, by expressing

\[
V_{J-2} \oplus W_{J-2} = V_{J-1}
\]

\[
\vdots
\]

\[
V_{J-1} \oplus W_{J-1} = V_j
\]

\[
\vdots
\]

\[
V_0 \oplus W_0 = V_1
\]

we obtain recursive formulae for the coefficients \( \{c_{0k}, d_{jk}, i \leq j \leq J-1\} \), in terms of the coefficients \( \{c_{jk}\} \), given by

\[
c_{j-1,l} = \frac{1}{\sqrt{2}} \sum_{r} c_{J,r} a_r - 2l
\]

\[
d_{j-1,l} = \frac{1}{\sqrt{2}} \sum_{r} c_{J,r} b_r - 2l
\]

for \( 0 \leq j \leq J \). The inverses of these formulae can be derived in a similar way. We simply consider

\[
V_{J-1} \oplus W_{J-1} = V_j
\]

and suppose that we know the coefficients of the expression in \( V_{j-1} \) and \( W_{j-1} \), and we want to determinate the coefficients in the expression in \( V_j \). Using a similar argument, we obtain the inductive formula

\[
c_{j,k} = \frac{1}{\sqrt{2}} \sum_{l} c_{j-1,l} a_{l-2k} + d_{j-1,l} b_{l-2k}
\]

for \( j = 1, \ldots, J \).
1.6. The Mallat algorithm for periodic data

The Mallat algorithm gives a linear mapping

\[ V_J^M \rightarrow V_0 \oplus W_0 \oplus ... \oplus W_{J-1} \]

and its inverse. These spaces are all infinite dimensional. We want to consider a special case where we replace each space \( V_j \) and \( W_j \) by its periodic analogue, thereby providing a finite-dimensional version of the Mallat algorithm, which is what is needed in practice. We define the periodic scaling and wavelet spaces with period \( L \) as

\[ V^L_j = \{ \sum c_{j,k} \phi_{jk} \mid c_{j,k} = c_{j,k+L}, k \in \mathbb{Z} \} \]

\[ W^L_j = \{ \sum d_{j,k} \psi_{jk} \mid d_{j,k} = d_{j,k+L}, k \in \mathbb{Z} \} \]

if we assume that \( L \) is even, then the Mallat algorithm maps naturally:

\[ \downarrow M \quad \downarrow H \oplus G \]

\[ V^L_{j-1} \oplus W^L_{j-1} \approx \mathbb{R}^{L/2} \oplus \mathbb{R}^{L/2} \]

and we see that the linear mapping \( M \) can be represented by a pair of matrices \( H \) and \( G \), which are illustrated for the case that \( g = 2 \) and \( L = 8 \):

\[ H = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ h_2 & h_3 & 0 & 0 & 0 & 0 & h_0 & h_1 \end{pmatrix} \]

and \( G \) will be similar with entries \( \{g_k\} \) replacing \( \{h_k\} \). Here we assume that the wavelet matrix \( A \) has the form

\[ A = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \end{pmatrix} = \sqrt{2} \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ g_0 & g_1 & g_2 & g_3 \end{pmatrix} \]

since the vectors \( (h_k) \) and \( (g_k) \) are what appear explicitly in the Mallat algorithm. The inverse Mallat algorithm will have a similar matrix representation.

**Observation.** The implementation of the Mallat algorithm is not carried out by matrix multiplication, but by a convolution with the vectors \( h = (h_0, h_1, h_2, h_3) \) and \( g = (g_1, g_2, g_3, g_4) \) followed by a downsampling operator. In this case, we only take the even terms. This is schematically described in Figure 3.
Let $L = 2K$ and let $s_0, \ldots, s_{2K-1}$ a sequence of numbers, then
\[
H(s_0, \ldots, s_{2K-1}) = (sl_0, \ldots, sl_{K-1}) \in \mathbb{R}^K \cong V_{j-1} \quad \text{and}
\]
\[
G(s_0, \ldots, s_{2K-1}) = (sh_0, \ldots, sh_{K-1}) \in \mathbb{R}^K \cong W_{j-1},
\]
the low-pass and high-pass outputs at the Mallat mapping $M$, respectively, are calculated as follows. Assume $g = 2$, then
\[
sl_0 = \sum_{l=0}^{3} s_lh_l \quad \quad \quad sh_0 = \sum_{l=0}^{3} s_lg_l
\]
\[
sl_1 = \sum_{l=2}^{5} s_lh_{l-2} \quad \quad \quad sh_1 = \sum_{l=2}^{5} s_lg_{l-2}
\]
\[
\vdots
\]
\[
sl_{K-1} = \sum_{l=2K-2}^{2K+1} s_lh_{l-2K-2} \quad \quad \quad sh_{K-1} = \sum_{l=2K-2}^{2K+1} s_lg_{l-2K-2}
\]

**Observation.** For the last sum, we extend the sequence periodically by setting $s_0 = s_{2K}$ and $s_1 = s_{2K+1}$.

**Definition.** Let $A$ be a wavelet matrix of rank 2 $(x_0, \ldots, x_{2N-1}) \in \mathbb{R}^{2N}$. Let $J$ be an integer such that $1 \leq J \leq N$. The **discrete wavelet transform** (DWT) of order $J$ defined by $A$ is a linear mapping
\[
\mathbb{R}^{2N} \xrightarrow{N} \mathbb{R}^{2^{N-1}} \oplus \mathbb{R}^{2^{N-2}} \oplus \cdots \oplus \mathbb{R}^{2^{N-J}} \oplus \mathbb{R}^{2^{N-J}}
\]
\[
V_N^{L} \xrightarrow{N} W_{N-1}^{L} \oplus W_{2^{N-2}}^{L} \oplus \cdots \oplus W_{2^{N-J}}^{L} \oplus W_{2^{N-J}}^{L} \oplus V_{2^{N-J}}^{L}
\]
where we identify the vector spaces $\mathbb{R}^{2N}$ with the periodic scaling and wavelet spaces of periods $2^j$ as indicated above. The mapping $M$ is given by the Mallat algorithm. The scheme is sketched in Figure 3.

**Observation.** It is not hard to see that the computational complexity of the discrete wavelet transform is $O(N)$. Indeed, at every level the convolution involves $L$ products and $L$ summands, so the total number of operations at any level is bounded by $2LN$. Since at every level we operate on half the number of points as at the previous level, and $(1 + \frac{1}{2} + \ldots + \frac{1}{2^J}) < 2$, we see that we at most double the constant by going to a finite number $J$ of levels.

### 1.7. Wavelet image compression

Today, the most successful general image compression method is based on the two dimensional product basis constructed from the Daubechies wavelet matrix $D_3$ and their biorthogonal variations. Recall that $D_3$ provides a low-pass representation of polynomials of degree less than 3. From this it follows that the product basis provides a low-pass expansion of polynomial functions of two variables, say $x$ and $y$, of degree less than 3. These quadric surfaces are good models for light reflected from smooth surfaces such as walls, or the cheeks and forehead of a face. The ability to represent reflected light energy that produces quadric surfaces in a digital image means that this transform is effective in concentrating a large fraction of the image energy in a small number of low-pass ("near dc") transform coefficients. The Figure displays the scaling and wavelet functions $\varphi(x)$ and $\psi(x)$.
The filters that correspond to these basis functions transform the original image into four components: the low-low, low-high, high-low and high-high parts of the transform.

Each of these outputs contains one quarter as many coefficients as the input image, so the total number of output coefficients is the same as the number of input coefficients. This fact makes it possible to display the transformed output as an image also by scaling all outputs to fall in the range of the input pixel values—say the range 0 to 255 for an 8 bit gray-scale image. The four images in the four subsquares of Figure 5 were constructed in this way. They illustrate the first stage in applying the wavelet transform for compression, and correspond to the low-low, low-high, high-low, and high-high pass components of the original image (which looks like the low-low quadrant in the upper left).

1.8. Wavelet channel coding

The user of a digital communication system will primarily be concerned with the bit rate and the error rate of the communication channel. The communication channel is controlled by allocating power and bandwidth, both of which are precious resources.

Every communication channel experiences noise. Additive white Gaussian noise (AWGN) is often a good first approximation because it results from specific applications. The communication system designer would like to achieve the objectives of high bit rate and low error rate with a minimum of power consumption and bandwidth requirements. These objectives are inconsistent; trade-offs must be made that prefer high bit rate to low error rate or low error rate to high bit rate; low power requirements to low bandwidth or low bandwidth to low power requirements. These trade-offs determine the type of modulation and error correction that will be selected.

Wavelet channel coding (WCC) provides a systematic conceptual and design method for making these trade-offs through software selection within a common chipset-based hardware framework.
The advantages of wavelet channel coding stem from the combination of the exact orthogonality of wavelet codewords, the spreading of message symbol information throughout the codeword symbols, and the ability to utilise soft decisions to decode wavelet channel coded symbols.

The main properties of WCC are summarised in the following list:

1. Wavelet Channel Coding is a new class of coding algorithms.
2. WCC offers a systematic and simple approach to the full range of coding problems, from low rate, power efficient codes to high rate, bandwidth efficient codes.
3. Wavelet Channel Coding can be applied in a block coding mode or in a trellis coding mode.
4. WCC codewords are strictly orthogonal.
5. WCC codewords have small correlations for non-codeword offsets.
6. WCC coding employs robust soft decision decoding.
7. WCC codes can be arbitrarily long.
(8) WCC codewords have desirable Hamming distance properties and provide good error correction capability.
(9) WCC codes can be efficiently coded and decoded with simple VLSI circuits.
(10) Synchronisation can be performed by correlation matching.
(11) Analysis and computer simulations show that WCC with arbitrary length wavelets achieves the AWGN performance of coherent BPSK\(^2\) When compared under a constant channel rate criterion, WCC provides coding gains of approximately \(10 \log_2 (mg + 1)\) dB. On pulsed interference and flat fading channels, WCC outperforms BPSK with gains that are dependent on the wavelet sequence length.

**Example.** Wavelet Channel Coding for \(m = 2\)

For the purpose of illustration, let a message

\[ x_1, x_2, ..., x_n, ... \]

consist of a sequence of bits represented by the signed units \(+1, -1\). Let

\[
A = \begin{pmatrix}
ad_0 & a_1 & \ldots & a_{2g-1} \\
b_0 & b_1 & \ldots & b_{2g-1}
\end{pmatrix}
\]

be a wavelet matrix of rank 2 and genus \(g\) and let \(a\) and \(b\) denote the scaling and wavelet vectors of this wavelet matrix, respectively (first and second rows of \(A\), or, in engineering language, the low and high pass filters of the 2-band filter bank \(A\)). Because the rows of \(A\) are mutually orthogonal when translated by steps of two, we can take advantage of the orthogonality of the filters (vectors) \(a\) and \(b\) to code at a rate of two message bits per \(2k\) clock pulses, where \(1 < k < \log_2 2g\). Message bits with odd sequence number will be coded using the waveform \((a_0, ..., a_{2g-1})\) and message bits with even sequence number will be coded using the waveform \((b_0, ..., b_{2g-1})\).

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>2g</th>
<th>2g + 1</th>
<th>2g + 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(x_1d_0)</td>
<td>(x_1a_1)</td>
<td>(x_1a_2)</td>
<td>(x_1a_3)</td>
<td>(x_1a_{2g-1})</td>
<td>(x_1a_{2g-1})</td>
</tr>
<tr>
<td>(x_2b_0)</td>
<td>(x_2b_1)</td>
<td>(x_2b_2)</td>
<td>(x_2b_3)</td>
<td>(x_2b_{2g-1})</td>
<td>(x_2b_{2g-1})</td>
<td>(x_2b_{2g-1})</td>
</tr>
<tr>
<td>(x_3d_0)</td>
<td>(x_3a_1)</td>
<td>(x_3a_2)</td>
<td>(x_3a_3)</td>
<td>(x_3a_{2g-2})</td>
<td>(x_3a_{2g-1})</td>
<td>(x_3a_{2g-1})</td>
</tr>
<tr>
<td>(x_4b_0)</td>
<td>(x_4b_1)</td>
<td>(x_4b_2)</td>
<td>(x_4b_3)</td>
<td>(x_4b_{2g-2})</td>
<td>(x_4b_{2g-1})</td>
<td>(x_4b_{2g-1})</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>(y_1)</td>
<td>(y_2)</td>
<td>(y_3)</td>
<td>(y_4)</td>
<td>(y_{2g})</td>
<td>(y_{2g+1})</td>
<td>(y_{2g+2})</td>
</tr>
</tbody>
</table>

Table 1. Wavelet channel coding example.

The coding procedure is illustrated in Table 1 for a system with maximally overlapped wavelet sequences. This corresponds to the case where the code rate is one information bit per WCC symbol (i.e. \(k = 1\) above). The sequence of clock pulses is arranged above the upper rule of the table, and the sequence of transmitted symbols \(y_n\) is shown below the lower rule of the table. The encoding of each message bit \(x_n\) is shown in the \(n\)th row below the upper rule. Note that for odd \(n\) the encoding begins on the \(n\)th clock pulse and that for even \(n\) the encoding begins on the \((n - 1)\)th clock pulse. The symbol \(y_n\) is equal to the sum of the \(n\)th column of encoded message bit values and is therefore not restricted to the values \(+1\). The formula for computing the symbol \(y_n\) to be transmitted at the \(n\)th clock pulse is

\[ y_n = \sum_k \{x_{2k+1}a_{n-2k-1} + x_{2k+2}b_{n-2k-1}\} \]

\(^2\)BPSK refers to Binary Phase Shift Keying and it is a modulation technique.
Some observations are in order:

(1) It is evident that code rates as small as $1/g$ can be achieved by varying the overlap of the wavelet sequences; in the limit where the code rate is $1/g$, the wavelet sequences are non overlapped but adjacent.

(2) When flat wavelet matrices are used, the WC encoded symbols that are generated by the WCC algorithm are binomially distributed.
Chapter 2
Wallpaper groups

In this chapter we present the theory of wallpaper groups that we need in the next chapter. In our theoretical approach we follow [37]. First, we review the isometry group of the plane and in view of our needs. After that, we define the wallpaper groups and their lattices. Next, we state the fact (theorem [5]) that there are exactly 17 different types of wallpaper groups under isomorphism [49]. Finally we explain the properties, summarized in Table 1 that single out each of these groups from the others.

2.1. Euclidean plane isometries

Definition. An Euclidean plane isometry is a map $M : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$d(p, q) = d(Mp, Mq)$$

for any two points $p, q \in \mathbb{R}^2$, where $d(p, q)$ is the Euclidean distance between them.

Isometries of $\mathbb{R}^2$ form a group, to be denoted $\text{Isom}(\mathbb{R}^2)$, under the composition law. If $F \subset \mathbb{R}^2$, we will set $\text{Isom}(F)$ to denote the subgroup of $\text{Isom}(\mathbb{R}^2)$ of the isometries $M$ that leave $F$ invariant, that is, such that $M(F) = F$.

Example. If $\tilde{M}$ is an orthogonal $2 \times 2$ matrix and $v \in \mathbb{R}^2$, then the map defined by

$$M(p) = \tilde{M}p + v$$

is an isometry. Note that if $\tilde{M}$ is orthogonal, then

$$d(Mp, Mq)^2 = (M(p) - M(q))^2 = (\tilde{M}(p - q))^2 = (p - q)^2 = d(p, q)^2.$$

A basic result in elementary Euclidean geometry is that the map $(\tilde{M}, v) \mapsto M$, where $M$ is given as in the preceding example, is a bijection. Given an isometry $M$, the $\tilde{M}$ uniquely determined by this bijection is said to be the associated linear map of $M$. Since $\tilde{M}$ is orthogonal, we know that $\det(\tilde{M}) = \pm 1$. In the case $+1$ (respectively $-1$), $\tilde{M}$ preserves (reverses) the orientation of $\mathbb{R}^2$ and we say that $M$ is a direct (reverse) isometry.

In next examples we will consider four special types of isometry that will play a key role in the sequel.
Example. Given a vector \( v \in \mathbb{R}^2 \), the translation defined by \( v \), and denoted \( T_v \), is defined by
\[
T_v(p) = p + v.
\]
It is clear that it is an isometry (it can be written \( T_v(p) = I_2 p + v \) and \( I_2 \) is orthogonal). If \( v = 0 \), then \( T_v = Id \), the identity map, but if \( v \neq 0 \), then \( T_v \) has no fixed points, for \( p + v = p \) for a single \( p \) implies \( v = 0 \).

The map \( \mathbb{R}^2 \to \text{Isom}(\mathbb{R}^2) \) given by \( v \mapsto T_v \) is a group homomorphism, as \( T_v \circ T_w = T_{v+w} \). Moreover, this map is injective, for \( T_v = Id \) only occurs for \( v = 0 \). It follows that the translations of \( \mathbb{R}^2 \) form a subgroup of \( \text{Isom}(\mathbb{R}^2) \), to be denoted \( \mathbb{T}(\mathbb{R}^2) \), and that the map \( \mathbb{R}^2 \to \mathbb{T}(\mathbb{R}^2) \) is an isomorphism.

If \( F \subset \mathbb{R}^2 \), we define \( \mathbb{T}(F) \) as the set of translations of \( \mathbb{R}^2 \) that leave \( F \) invariant. It is a subgroup of \( \mathbb{T}(\mathbb{R}^2) \) which corresponds to the subgroup formed by the \( v \in \mathbb{R}^2 \) such that \( F + v = F \).

Since \( \det(I_2) = 1 \), any translation is a direct isometry.

Example. For any real number \( \theta \), the matrix
\[
R_\theta = R_{0,\theta}(p) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} [p]
\]
is orthogonal and \( \det(R(\theta))=1 \). Thus the map
\[
p \mapsto R_\theta p,
\]
represents a rotation of amplitude \( \theta \) about the origin, is a direct isometry. More generally, if \( c \in \mathbb{R}^2 \), then the map
\[
p \mapsto c + R_\theta(p-c) = R_\theta p + v, \quad (v = c - R_\theta c = (I_2 - R_\theta)c)
\]
is the rotation about \( c \) of amplitude \( \theta \).

A rotation around \( c \) can be accomplished by first translating \( c \) to the origin, then performing the rotation about the origin, and finally translating the origin back to \( c \):
\[
R_{c,\theta} = T_c \circ R_{0,\theta} \circ T_{-c}
\]
Indeed, \( c + R_\theta(p-c) = T_c(R_\theta(T_{-c}p)) = (T_c R_\theta T_{-c})p \).

Example. Let \( L \subset \mathbb{R}^2 \) be a line. The reflection with respect to \( L \), \( F_L \), is defined as follows. Given a point \( p \), \( \tilde{p} = F_L(p) \) is the point on the line through \( p \) that is perpendicular to \( L \) and such that the midpoint of the segment \( p \tilde{p} \) lies on \( L \). In the case that \( L \) is the \( x \)-axis, for instance, then \( \tilde{p} = \begin{pmatrix} x \\ -y \end{pmatrix} \) if \( p = \begin{pmatrix} x \\ y \end{pmatrix} \). More generally, if \( u \) is a unit vector parallel to \( L \), and \( u' \) is the unit vector such that \( u' \) is perpendicular to \( u \) and \( \det(u,u') = 1 \), then
\[
F_L(p) = p_0 + \tilde{M}(p - p_0)
\]
where \( \tilde{M} = (u, -u') \) and \( p_0 \) is an arbitrary point on \( L \). Since \( \det(\tilde{M}) = -1 \), we conclude that reflections are reverse isometries.

Example. Let \( L \subset \mathbb{R}^2 \) and \( w \) a non-zero vector parallel to \( L \). Then the glide reflection \( G_{L,w} \) is defined as
\[
G_{L,w}(p) = T_w(F_L(p)) = F_L(p) + w.
\]
In other words, \( G_{L,w} \) is the composition of the reflection \( F_L \) and the translation defined by the vector \( w \).
Remark that it is also true that \( G_{L,w}(p) = F_L(p + w) \), as follows from the fact that \( \tilde{M}(w) = w \).
Theorem 4. Every isometry of the plane, other than the identity, is either a translation, a rotation, a reflection, or a glide-reflection. If it has no fixed points, it is a translation or a glide-reflection according to whether the isometry is direct or reverse. If it has fixed points, it is rotation if it has a unique fixed point or a reflection if it has more than one fixed point (in which case it has a line of fixed points).

The proof of this theorem can be found in elementary Geometry texts such as [59] [3] [51]. Notice that an isometry with a unique fixed point is direct, as it is a rotation, whereas an isometry with a line of fixed points is reverse; it is a reflection.

2.2. Wallpaper groups

A lattice of $\mathbb{R}^2$ is any subset $T$ that has the form

$$T = \{ nu + mv : n, m \in \mathbb{Z} \}$$

with $u, v \in \mathbb{R}^2$ linearly independent. In this case $u, v$ is said to be a basis of the lattice. The parallelogram defined by $u, v$, which will be denoted $[u, v]$, is said to be a cell (or a tile) of the lattice.

If $T$ is lattice, then it contains

1. a nonzero vector of minimal length, and
2. only finitely many vectors inside any circle.

Given the lattice $T$, there are infinitely many bases $u', v' \in T$, that is, such that

$$T = \{ nu' + mv' : n, m \in \mathbb{Z} \}.$$

These bases have the form $v' = au + bv, w' = cu + dv$, with $a, b, c, d \in \mathbb{Z}$ and $ab - cd = \pm 1$.

A reduced cell is a cell $[u', v']$ such that $v'$ has minimal length, the length of $u'$ is as close as possible to the length of $u$, and the angle $\hat{u'}, v'$ is acute (if obtuse, replace $u'$ by $-u'$). Henceforth we will assume that the basis $u, v$ is chosen so that the cell $[u, v]$ is reduced and cell will mean reduced cell.

If we let $a$ and $b$ denote the lengths of $u$ and $v$, and $\alpha = \hat{u}, \hat{v}$, then we have the following types of cell (see Figure 1):

- **Oblique**: $a > b$ and $\alpha < 90^\circ$.
- **Rectangular**: $a > b$ and $\alpha = 90^\circ$.
- **Rhombic**: $a = b$ and $\alpha \neq 90^\circ, 60^\circ$.
- **Square**: $a = b$ and $\alpha = 90^\circ$.
- **Hexagonal**: $a = b$ and $\alpha = 60^\circ$.

![Oblique Rectangular Rhombic Square Hexagonal](image.png)

**Fig. 1.** Lattice cell types.
**Definition.** A subset \( W \) of \( R^2 \) is a wallpaper pattern if the translation subgroup \( T(W) \) of the symmetry group \( \text{Sym}(W) \) is a lattice. The symmetry group of a wallpaper pattern is said to be a wallpaper group or a plane crystallographic group.

An important property of the wallpaper groups is the crystallographic restriction theorem. This result asserts that if a wallpaper group contains a rotation other than the identity, then the order of this rotation can only be 2, 3, 4, or 6 (the identity has order 1).

The existence of rotations is closely related with the cell type. If we let \( r \) denote the highest order of a rotation, then the relation with the cell type can be summarized as follows:

- If there are no symmetries other than translations (in particular \( r = 1 \)) or if \( r = 2 \), the cell is a parallelogram (oblique type), which can be special (rectangle, rhombus or square).
- If \( r = 3 \) or \( r = 6 \), the cell is hexagonal.
- If \( r = 4 \), the cell is a square.

If \( r = 1 \), but with reflections or glide reflections, then the possible cell types are as follows:

- If there is a reflection or a glide reflection, but not both, the cell type is rectangular, which as a special case can be a square.
- If there is a reflection and a glide reflection, the cell type is rhombic, which as a special case can be a square.

### 2.3. Wallpaper group notations

In this subsection we present the two more standard wallpaper notations.

#### 2.3.1. Crystallographic notation.
Crystallography has 230 space groups to distinguish, far more than the 17 wallpaper groups, but many of the symmetries in the groups are the same.

First of all, we choose a basis \( \{T_u, T_v\} \) for the translation lattice. We consider the direction of \( T_u \) to be the \( x \)-axis. The full name consists of four symbols. The first symbol represents the lattice type: \( p \) for primitive and \( c \) for centered (or rhombic). The second symbol is the largest order of a rotation. The third symbol is either an \( m \), \( g \), or 1. An \( m \) (respectively \( g \)) means that there is a reflection line (respectively glide reflection line but not a reflection line) perpendicular to the \( x \)-axis while a 1 means there is no line of either type. Finally, the fourth symbol is also either an \( m \), a \( g \), or a 1. In this case an \( m \) (resp. \( g \)) represents a reflection line (resp. glide reflection line) at an angle \( \alpha \) with the \( x \)-axis, the angle depending on the largest order of rotation as follows:

- \( \alpha = \pi \) for \( n = 1, 2 \).
- \( \alpha = \frac{\pi}{2} \) for \( n = 3, 6 \).
- \( \alpha = \frac{\pi}{4} \) for \( n = 4 \).

For example, the group name \( p3m1 \) represents a group with a \( \frac{2\pi}{3} \) rotation, a reflection line perpendicular to the \( x \)-axis, and no reflection or glide line at an angle of \( \frac{\pi}{4} \) with the \( x \)-axis. However, in the group \( p31m \), we...
have the same rotation, but no reflection or glide line perpendicular to the $x$-axis, while there is a reflection line at an angle of $\frac{\pi}{3}$ with the $x$-axis.

### 2.3.2. Orbifold notation.

Orbifold notation or Conway notation for wallpaper groups, introduced by John Horton Conway, is based not on crystallography, but on topology. The notation takes the symmetries in the orbifold of the tiling. In the present work we do not talk about orbifolds.\(^1\)

A digit, $n$, indicates a centre of $n$-fold rotation. By the crystallographic restriction theorem, $n$ must be 2, 3, 4, or 6.

An asterisk, $\ast$, indicates a reflection. It interacts with the digits as follows:

- Digits before $\ast$ denote centres of pure rotation (cyclic).
- Digits after $\ast$ denote centres of rotation with reflection through them.

A cross, $\times$, occurs when a glide reflection is present. Reflections combine with lattice translation to produce glides, but those are already accounted for so we do not notate them.

The "no symmetry" symbol, $\circ$, stands alone, and indicates we have only lattice translations with no other symmetry.

Consider the group denoted in crystallographic notation by $cmm$; in Conway’s notation, this will be $2\ast 22$. The 2 before the $\ast$ says we have a 2-fold rotation centre with no reflection through it. The $\ast$ itself says we have a reflection. The first 2 after the $\ast$ says we have a 2-fold rotation centre on a reflection axis. The last 2 says we have an independent second 2-fold rotation centre on a reflection axis, one that is not an image of a center of the first kind by any symmetry.

The group denoted by $pgg$ will be $22\times$. We have two pure 2-fold rotation centres, and a glide reflection axis. Contrast this with $pmg$, Conway $22\ast$, where crystallographic notation mentions a glide that it is implicit in the other symmetries.

In the table\(^1\) there is the correspondence between the different wallpaper groups in both notations.

### 2.4. The 17 wallpaper groups

This subsection we state the well known theorem that there are exactly 17 different wallpaper group under isomorphism and we present an explanation of every group.

**Theorem 5.** There exactly 17 non-isomorphic wallpaper groups.

We do not prove this theorem here but there is a beautiful and elementary proof, due to Schwarzenberger, that uses only basic trigonometry and group theory in [49].

In the following lines we present the properties of each group. At the end, table\(^1\) summarising the 17 wallpaper groups and their properties. After that, there is the figure\(^2\) where are presented the main cell and the symmetries of each group.

---

\(^1\)To know more about orbifold see [42].
2.4.1. **p1.** This is the simplest symmetry group. It consists only of translations. There are neither reflections, glide-reflections, nor rotations. The two translation axes may be inclined at any angle to each other.

2.4.2. **pm.** This group contains reflections. The axes of reflection are parallel to one axis of translation and perpendicular to the other axis of translation. The lattice is rectangular. There are neither rotations nor glide reflections.

2.4.3. **pg.** This group contains glide reflections. The direction of the glide reflection is parallel to one axis of translation and perpendicular to the other axis of translation. There are neither rotations nor reflections.

2.4.4. **cm.** This group contains reflections and glide reflections with parallel axes. There are no rotations in this group. The translations may be inclined at any angle to each other, but the axes of the reflections bisect the angle formed by the translations, so the lattice is rhombic.

2.4.5. **p2.** This group differs only from p1 in that it contains π rotations, that is, rotations of order 2. As in all symmetry groups there are translations, but there are no reflections nor glide reflections. The two translations axes may be inclined at any angle to each other.

2.4.6. **pgg.** This group contains no reflections, but it has glide-reflections and π rotations. There are perpendicular axes for the glide reflections, and the rotation centers do not lie on the axes.

2.4.7. **pmg.** This group contains reflections, and glide reflections which are perpendicular to the reflection axes. It has rotations of order 2 on the glide axes, halfway between the reflection axes.

2.4.8. **pmm.** This symmetry group contains perpendicular axes of reflection, with π rotations where the axes intersect.

2.4.9. **cmm.** This group has perpendicular reflection axes, as does the group pmm, but it also has rotations of order 2. The centers of the rotations do not lie on the reflection axes.

2.4.10. **p3.** This is the simplest group that contains a \( \frac{2\pi}{3} \)-rotation, that is, a rotation of order 3. It has no reflections or glide reflections.

2.4.11. **p31m.** This group contains reflections (whose axes are inclined at \( \frac{\pi}{3} \) to one another) and rotations of order 3. Some of the centers of rotation lie on the reflection axes, and some do not. There are some glide-reflections.

2.4.12. **p3m1.** This group is similar to the last in that it contains reflections and order-3 rotations. The axes of the reflections are again inclined at \( \frac{\pi}{3} \) to one another, but for this group all of the centers of rotation do lie on the reflection axes. There are some glide-reflections.

2.4.13. **p4.** This group has a \( \frac{\pi}{4} \) rotation, that is, a rotation of order 4. It also has rotations of order 2. The centers of the order-2 rotations are midway between the centers of the order-4 rotations. There are no reflections.
2.4.14. **p4g.** Like \( p4 \), this group contains reflections and rotations of orders 2 and 4. There are two perpendicular reflections axis passing through each order 2 rotation. However, the order 4 rotation centers do not lie on any reflection axis. There are four directions of glide reflection.

2.4.15. **p4m.** This group also has both order 2 and order 4 rotations. This group has four axes of reflection. The axes of reflection are inclined to each other by \( \frac{\pi}{4} \) so that four axes of reflection pass through each order 4 rotation center. Every rotation center lies on some reflection axes. There are also two glide reflections passing through each order 2 rotation, with axes at \( \frac{\pi}{4} \) to the reflection axes.

2.4.16. **p6.** This group contains \( \frac{\pi}{3} \) rotations, that is, rotations of order 6. It also contains rotations of orders 2 and 3, but no reflections or glide-reflections.

2.4.17. **p6m.** This complex group has rotations of order 2, 3, and 6 as well as reflections. The axes of reflection meet at all the centers of rotation. At the centers of the order 6 rotations, six reflection axes meet and are inclined at \( \frac{\pi}{6} \) to one another. There are some glide reflections.

<table>
<thead>
<tr>
<th>Crist. notation</th>
<th>Orbiforl notation</th>
<th>Lattice</th>
<th>Rotation Order</th>
<th>Reflection Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p1 )</td>
<td>( \circ )</td>
<td>parallelogram</td>
<td>1</td>
<td>none</td>
</tr>
<tr>
<td>( pm )</td>
<td>( \ast )</td>
<td>rectangle</td>
<td>1</td>
<td>parallel</td>
</tr>
<tr>
<td>( pg )</td>
<td>( \times )</td>
<td>rectangle</td>
<td>1</td>
<td>none</td>
</tr>
<tr>
<td>( cm )</td>
<td>( \ast \times )</td>
<td>rhombus</td>
<td>1</td>
<td>parallel</td>
</tr>
<tr>
<td>( p2 )</td>
<td>2222</td>
<td>parallelogram</td>
<td>2</td>
<td>none</td>
</tr>
<tr>
<td>( pgg )</td>
<td>22\times</td>
<td>rectangle</td>
<td>2</td>
<td>none</td>
</tr>
<tr>
<td>( pmg )</td>
<td>22\ast</td>
<td>rectangle</td>
<td>2</td>
<td>parallel</td>
</tr>
<tr>
<td>( pmm )</td>
<td>2\ast22</td>
<td>rectangle</td>
<td>2</td>
<td>90°</td>
</tr>
<tr>
<td>( cmm )</td>
<td>2\ast22</td>
<td>rhombus</td>
<td>2</td>
<td>90°</td>
</tr>
<tr>
<td>( p3 )</td>
<td>333</td>
<td>hexagonal</td>
<td>3</td>
<td>none</td>
</tr>
<tr>
<td>( p31m )</td>
<td>3\ast3</td>
<td>hexagonal</td>
<td>3'</td>
<td>60°</td>
</tr>
<tr>
<td>( p3m1 )</td>
<td>*333</td>
<td>hexagonal</td>
<td>3'</td>
<td>60°</td>
</tr>
<tr>
<td>( p4 )</td>
<td>442</td>
<td>square</td>
<td>4</td>
<td>none</td>
</tr>
<tr>
<td>( p4g )</td>
<td>4\ast2</td>
<td>square</td>
<td>4''</td>
<td>90°</td>
</tr>
<tr>
<td>( p4m )</td>
<td>*442</td>
<td>square</td>
<td>4'</td>
<td>45°</td>
</tr>
<tr>
<td>( p6 )</td>
<td>632</td>
<td>hexagonal</td>
<td>6</td>
<td>none</td>
</tr>
<tr>
<td>( p6m )</td>
<td>*632</td>
<td>hexagonal</td>
<td>6</td>
<td>30°</td>
</tr>
</tbody>
</table>

Table 1. Properties of all 17 wallpaper groups.

\( ^\prime = \) all rotation centers lie on reflection axes.

\( ^\prime\prime = \) not all rotation centers on reflection axes.
Fig. 2. The 17 wallpaper groups main cell with all their rotational centers, reflections and glide reflections.
Chapter 3
Phonetic techniques

3.1. Introduction

In this chapter we assemble key materials about phonetics that are needed for the application in Chapter 6. These materials are basically all the terms used in speech processing explained in section 3.2. We also present a simple version of some modern techniques used in speech signals. First, in sections 3.3 to 3.6 we explain a typical speech recogniser. After that, in the last sections, we consider speech synthesis and explain how a concatenative synthesiser works. The principal references used in this chapter are \[26,15,28\].

Nowadays the speech recognition systems are best understood by analysing three aspects separately (see \[28,26\]): feature extraction, acoustic model and language model. Most techniques to extract features from the signals are aimed at improving the audio quality before recognition, as a way of strengthening the recognition process. These methods are basically Mel Frequency Cepstral Coefficients (MFCC) (See section 3.6) and Perceptual Linear Prediction (PLP) \[10,24,26,22,15,36,28\]. Other techniques that have been used recently for feature extraction implement wavelet techniques. These systems use the local properties of wavelet transform to extract the energies of each subband of frequencies and use these energies as feature signatures \[32,13\].

The second aspect is the acoustic model, whose role is to create statistical representations of the phonemes and stitch them together to form words. Most common recognisers use HMM models (See section 3.4). There are several good references that use this method, such as \[27,16,26,36,15,28\]. Other techniques use artificial neural networks (see Appendix A) to classify phonemes and create words from them \[30,56\].

Finally, the third aspect tries to identify the most probable sequence of words in the speech signal. The most common approach is the N-gram model (See section 3.5). Some works that use this model are \[57,26,15,28\].

There are mainly three types of speech synthesisers depending on how the waveform is created. These types are concatenative synthesisers, formant synthesisers and articulatory synthesisers.

Concatenative synthesis generates a speech waveform by concatenating a sequence of diphones (See section 3.10) from a database. After that, it is necessary to adjust the waveform in order to obtain speech with an acceptable prosody. We refer to \[53,52,26\] for description of some modern synthesisers.
Formant synthesis tries to mimic the speech spectrogram by generating the formants (see section 3.2) of the different phonemes. The most famous synthesizers of this type are Klatt’s formant synthesiser and its successors, including [1]. To know more about them see [26].

Finally, articulatory synthesis attempts to generate speech by modeling the vocal tract as an open tube (cf. [12, 26]).

3.2. Phonetic sounds

Humans produce sounds called phonemes by expelling air from the lungs that resonates among the vocal tract (laryngeal cavity, the pharynx, the oral cavity, and the nasal cavity). Phonemes are usually divided into consonants and vowels. Consonants are usually created by blocking or restricting the airflow from the lungs. Consonants can be voiced or unvoiced\(^1\). Vowels have less obstructions and for our purposes they are voiced.

In this section we will describe some properties of the speech signals mostly related to the frequency domain.

First, we start by describing the bandwidth of sound signals. For practical purposes, the sounds which can be detected by human ears are below 10000 Hz. So, sampling rates used in speech processing are of the order of 16000 Hz\(^2\).

In terms of frequency, the most important term is fundamental frequency (often written as \(F0\)), which is the natural frequency of the vocal folds. This frequency, as we will see in section 3.9 depends on the utterance intonation.

![Fig. 1. Graph of the mel scale function.](image)

Another term related with the fundamental frequency is the pitch. The pitch of a sound is the acoustic perception of (the fundamental) frequency and hence it depends on the properties of human hearing. There are several variants of empirical functions to represent pitch in terms of frequency. One that is widely used is the mel scale.

\(^1\)If the vocal folds vibrate when the sound is created, the sound is said to be voiced and otherwise unvoiced.

\(^2\)Telephone signals use a bandwidth of 8000 Hz.
used, and which is sufficient for our needs, is the mel scale $m(f)$ (see Figure 1), which is defined by the formula

$$m(f) = 1127 \ln(1 + \frac{f}{700}).$$

This empirical function reflects the fact that equal increments of pitch correspond to equal ratios of frequency and is normalized so that $m(0) = 0$ and $m(1000) = 1000$.

Most speech signals are described using spectrograms. An spectrogram is a representation that how the frequency of a sound changes among the time. Zones with darken colours are zones with more power and zones with soft colors are frequencies with less power. An example can be seen in figure 2.

![Spectrogram](image)

Fig. 2. Spectrogram of a catalan word NEN. The x-axis is time in ms and the y-axis is frequency in Hz.

Each dark band in the figure is called formant. A formant is a frequency band that is amplified by the vocal tract. Since different phonemes has formants in different position they can be used as features in recognition such as in LPC models.

### 3.3. Markov models

Before explaining a recognition system we must explain Markov models which are use in it.

**Definition.** Let $\Omega$ be a finite set of states. A sequence $\{X_0, X_1, \ldots\}$ of random variables taking values in $\Omega$ is said to be a *Markov chain* if it satisfies the Markov condition:

$$P(X_n = i_n | X_0 = i_0, \ldots, X_{n-1} = i_{n-1}) = P(X_n = i_n | X_{n-1} = i_{n-1})$$

for all $n \geq 1$ and all $i_0, i_1, \ldots, i_n \in \Omega$.

**Definition.** The Markov chain is said *homogeneous* if, for all $n \geq 1$ and all $i, j \in \Omega$,

$$P(X_n = j | X_{n-1} = i) = P(X_1 = j | X_0 = i)$$
Markov chains will always be assumed homogeneous unless otherwise specified.

**Definition.** In this case, the transition matrix $Q$ of the chain is defined as the $|\Omega| \times |\Omega|$ matrix of all transition probabilities $Q(i, j) = a_{ij} = P(X_1 = j | X_0 = i)$. The probabilities $a_{ij}$ are called transition probabilities.

**Proposition.** The matrix $Q^n$ gives the law of $X_n$ for the chain starting at $X_0 = i$

$$P(X_n = j | X_0 = i) = Q^n(i, j)$$

**Definition.** A Hidden Markov Model is a Markov chain where each state has associated a probability density function $b_j(\theta_t)$ which determines the probability that this state generates $\theta_t$ at time $t$ (the model is hidden because any state could have emitted the observed value). $b_j$ are called observation probability.

**Observation.** The probability density function can be continuous or discrete. In our case (speech case) $b_j$ is usually a Gaussian distribution with a diagonal covariance matrix.

This model generates a sequence of observed states $\theta = \theta_0, \theta_1, ..., \theta_{T-1}$ because at every time $t$ only one state emits an observation. The set of all the sequences can be represented as routes via an scheme similar as in figure 3. In this scheme the $(j, t)$-th node corresponds to the hypothesis that the observation $\theta_t$ was generated by the state $j$. Two nodes $(i, t-1)$ and $(j, t)$ are connected if and only if $a_{ij} > 0$.

![Hidden Markov Model diagram](image)

Fig. 3. Hidden Markov Model, showing the finite state machine for the HMM (left), the observation sequence (top), and all the possible routes through the trellis (arrowed lines)

### 3.4. Speech recognition

As we have said before we will explain a simplified version of a speech recogniser based in a Hidden Markov model (HMM). To describe this recogniser we will follow [15].
First, this type of speech recogniser converts an input audio waveform into a sequence of fixed size acoustic vectors $Y_{1:N} = (y_1, ..., y_N)$. This process is called feature extraction (See section 3.6). After that, the decoder tries to find the most probably sequence of words $W_{1:M} = (w_1, ..., w_M)$ which have generated the input sequence $Y$. In other words, the decoder attempts to find

$$\hat{W} = \text{arg max}\{P(W|Y)\}$$

Now, as it is difficult to calculate this probability, we use the Bayesian rule to transform this equation into

$$\hat{W} = \text{arg max}\{P(Y|W)P(W)\} \quad (*)$$

As each spoken word $w$ is decomposed into a sequence of $K_w$ basic sounds called basic phonemes, the possibility of different pronunciations has to be taken into account. So, let $q^{(w)} = q_1, ..., q_{K_w}$ be a pronunciation of the word $w$. Thus, the likelihood $P(Y|W)$ can be expressed as

$$P(Y|W) = \sum_{Q} P(Y|Q)P(Q|W)$$

where the summation is over all valid pronunciation sequences for $W$ and $Q = q^{(w_1)}, ..., q^{(w_M)}$ is a particular sequence of pronunciations.

So if we suppose independence,

$$P(Q|W) = \prod_{l=1}^{M} P(q^{(w_l)}|w_l)$$

where each $q^{(w)}$ is a valid pronunciation for word $w_l$.

At this point, each base phoneme $q$ is represented by a continuous density HMM. Once taken the composite HMM $Q$ formed by all the base phonemes $q^{(w_1)}, ..., q^{(w_M)}$, the acoustic likelihood is given by the equation

$$P(Y|Q) = \sum_{\theta} P(\theta, Y|Q)$$

where $\theta = \theta_0, ..., \theta_{T+1}$ is a state sequence through the composite model and

$$P(\theta, Y|Q) = a_{\theta_0, \theta_1} \prod_{t=1}^{T} b_{\theta_t}(y_t)a_{\theta_t, \theta_{t+1}}$$

The acoustic model parameters $\lambda = [(a_{ij}), (b_j)]$ can be efficiently estimated using the forward-backward algorithm which is an example of expectation-maximisation algorithm described in [38].

**Observation.** The problem with decomposing the spoken words into a sequence of non-related phonemes is that it fails to capture the variations of this phoneme produced by the previous and following phonemes. So to eliminate this problem, more complex speech recognisers use triphonemes which is a unique model for every trio of phonemes.
3.5. N-gram approximation

In the equation (*) we have computed the first probability. In this section we will focus in the second term. The probability of word sequence \( W = w_1, \ldots, w_K \) is given by

\[
P(W) = \prod_{k=1}^{K} P(w_k | w_1, \ldots, w_{k-1})
\]

For recognition the previous product is truncated to \( N - 1 \) words to form an \( N \)-gram language model

\[
P(W) = \prod_{k=1}^{K} P(w_k | w_{k-N+1}, \ldots, w_{k-1})
\]

where \( N \) is typically in the range 2 – 4.

**Definition.** Given a finite alphabet \( S \) (in our case the set of all words), a *language on \( S \)* is a set of probability distributions \( P_n \) on the set of strings \( \Omega_n \) of length \( n \) for all \( n \geq 1 \).

To characterise a \( N \)-gram model we give three equivalent conditions

**Proposition.** Let \( \Omega_N \) be the space of strings \( a_1, \ldots, a_N \) of length \( N \), \( a_i \)'s being taken from a finite alphabet \( S \). Consider a probability distribution \( P : \Omega_N \rightarrow \mathbb{R}^+ \). Fix an integer \( n \geq 1 \). The following conditions are equivalent:

1. **Conditional factorisation.** \( P \) has the form

\[
P(a_1, \ldots, a_N) = P_0(a_1, \ldots, a_{n-1}) \prod_{k=0}^{N-n} P_1(a_{k+n} | a_{k+1}, \ldots, a_{k+n-1})
\]

for suitable \( P_0 \) and \( P_1 \).

2. **Exponential form.**

\[
P(a_1, \ldots, a_N) = \frac{1}{Z} \exp \left( - \sum_{\sigma \in \Omega_n} \lambda_{\sigma} \#occ(\sigma, a_1, \ldots, a_N) \right)
\]

for suitable \( \lambda_{\sigma} \)'s, where \( \Omega_n \) is the set of strings of length \( n \) and \( \#occ(\sigma, a_1, \ldots, a_N) \) denotes the number of occurrences of the string \( \sigma \) in \( a_1, \ldots, a_N \). Here we allow \( \lambda_{\sigma} = \infty \) in case no strings of positive probability contain \( \sigma \).

3. **Markov property.** For all \( I = (k + 1, \ldots, k + n - 1) \) of length \( n - 1 \), let \( a(I) = a_k + 1, \ldots, a_{k+n-1} \); then

\[
P(a_1, \ldots, a_N | a(I)) = P_1(a_1, \ldots, a_k | a(I)) P_2(a_k + n, \ldots, a_N | a(I))
\]

which means that a (before \( I \)) and a (after \( I \)) are conditionally independent given \( a(I) \).

Using the third equivalence, we can treat every \( I \) as an independent sequence and using Markov chains properties we can calculate the probabilities. The \( N \)-gram probabilities are estimated from training text by counting \( N \)-gram occurrences. For example, let \( C(w_k, w_{k-1}, w_{k-2}) \) represent the number of occurrences of the words \( w_{k-2}, w_{k-1}, w_k \), then

\[
P(w_k | w_{k-2}, w_{k-1}) \approx \frac{C(w_k, w_{k-1}, w_{k-2})}{C(w_{k-1}, w_{k-2})}
\]
3.6. Feature extraction

In this section we will explain the features of the speech recognition. There are a lot of possible feature representations but the most common is the mel frequency cepstral coefficients (MFCC). In this sections we will describe a method to extract these coefficients. To do it, we will follow [28].

First, the audio signal is convert into a digital signal. After that, the whole process is shown in figure 4.

Fig. 4. MFCC extraction process from a digital speech signal.

The first step is a high frequency filter to augment the high frequencies of the signal to make information of this part of the spectrum easier to detect. The preemphasis filter is a first order filter with an equation $y(n) = x(n) - \alpha x(n - 1)$ with $0.9 < \alpha < 1$.

As speech signals are generally non-stationary, it is common to extract the spectral features from a small part of the signal. If this part of the signal is sufficiently small, it can be supposed that the signal is stationary. To do it, the next step of the process is to run a window across the speech signal. This process can be characterized by three parameters. The width in time of the window, the offset between successive windowed samples (frames), and the shape of the window. In MFCC process it is often used the Hamming window, which decrease the value of the boundaries to zero to avoid discontinuities.

The goal of the upper branch of the figure 4 it is to calculate spectral features which are some values of the cepstral function. It is common to take as a features the first 12 cepstral values of the signal.

**Definition.** The Cepstral function $c(n)$ is formally defined as the inverse DFT of the log magnitude of the DFT of a signal $x(n)$. In other words,

$$c(n) = \sum_{k=0}^{N-1} \log \left( \sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N} \right)e^{i2\pi kn/N}$$

So, the upper branch of the scheme calculates is a sequence of process that their goal is to calculate the cepstral function and take its first 12 values.

**Observation.** As human hearing is less sensitive to higher frequencies (>1000Hz) than low frequencies, we pass the signal to a a bank of filters which model a mapping between hertz and mel scale to performance this human property.

Now, we have 12 cepstral coefficients for each frame. We also calculate in the lower branch the energy of the frame which can be defined as $E = \sum_{n=1}^{N} x(n)^2$.

---

3 See section 3.2
Finally, once we have these 13 values, we calculate the velocity and acceleration of these values and we have the final 39 features. The velocity is called delta features and the acceleration double deltas. A simple way to compute the delta features is to use differences between successive frames as

$$\Delta(c(t)) = \frac{c(t) - c(t - 1)}{2}$$

where each $c(t)$ is the sequence of features for the frame at time $t$.

So finally, the MFCC features are

- 12 cepstral coefficients
- 12 $\Delta$ cepstral coefficients
- 12 $\Delta\Delta$ cepstral coefficients
- 1 energy value
- 1 $\Delta$ energy value
- 1 $\Delta\Delta$ energy value

### 3.7. Text Normalisation

After explaining the speech recognition, we present a method to synthesize speech from text. The modern tasks of speech synthesis are called text-to-speech (TTS). These synthesis systems performs the mapping in two steps. First, it converts the text into phonemic representation (text analysis) and then converting this representation into a speech signal (waveform synthesis). In our approach, we present a concatenative TTS system because they are the most commercial one. In our approach we will follow [28]. Other examples of TTS systems are formant synthesizers and articulatory synthesizers.

First, to generate a phonetic representation of the text we first need to normalize the text in a variety of ways. First, we need to segment the text into separate utterances for synthesis. Second, we need to deal with non-standard words such as numbers, acronyms, abbreviations, etc. Dealing with these types of words requires a three step algorithm: identify potential non-standard words, classify these words and replace these words for an expansion of them. To do it, a dictionary table is used to make the replacement.

**Observation.** Some languages such as English has another type of problem, that is there words that have more than one pronunciation. They are called homograph. An example could be the word “live”: Do you live (/l ih v/) near a zoo with live (/l ay v/) animals?. This problem are often ignored in TTS systems but there is an algorithm to deal with it explained in [60].

### 3.8. Phonetic analysis

The next part of the process is to produce a pronunciation for each word in the normalized string of words from the original text. First every word is look up in a phonetic dictionary to catch its pronunciation. However, this type of dictionaries has pronunciations for about 120000 words. So, there are some words in the string that there are not in the dictionary.

---

4These type of synthesizers can be found in [26]
So, the remaining words must be created its pronunciation. This process is called *grapheme-to-phoneme conversion*. First, this process has a table named *letter-to-sound* where for every letter it has all its possible different phonemes. After that, the program tries to find the most probable phoneme sequence $P$ from a letter sequence $L$

$$P = \arg \max_{P} P(P|L)$$

To do it, it is often used $N$-grams in a similar ways as in section 3.5.

### 3.9. Prosodic analysis

The final step in phonetic representation is prosodic analysis. The word *prosody* refers to the intonational and rhythmic aspects of the language. This word refers to the acoustic features like tone, duration and energy of the sentence independently of the phoneme string. With different features one can reproduce different types of emotional meanings such as anger, happiness, etc. The three aspects of the prosody are *prosody structure*, *prosody prominence* and *tune*.

In spoken sentences there are some words that seems to group together and others that seems to be a pause between them. This structure is called *prosody structure* of the sentence. An example could be the sentence *I wanted to go to Barcelona* where the sentence seems to split into *I wanted — to go — to Barcelona*. Some implications of this aspect could be that the final vowel of a phrase is longer than usual, a pause is inserted after an intonation phrase, etc.

The models to deal with this problem are based on machine classifiers. So, a training set must be given to the classifier to train it. This training set is formed by a set of sentences where labels are putted to mark the boundaries of the prosody structure of the sentence. After that the classifier decide where are the breaks of the new text after training.

*Prosodic prominence* refers to that in sentences there are some words that sounds more important that others. This prominence can be detected by humans because these words are said louder, slower or it is a change in its frequency. In other words, we capture the prominence words because we associate a marked called *pitch accent*. Other words, can be less prominent that the normal ones such as function words like *of*, *and*, *it*, etc. So, we can sumarize it saying that there are four types of prominence which are *emphatic accent*, *pitch accent*, *unaccented* and *reduced*. However, common synthesisers has only two or three types of prominence.

To predict if a word must be accented or not some semantic knowledge has to be taken into account. This semantic knowledge could be if the word is new or not in the text, the quantity of information in the word, etc. For example, the fact of information in a word can be modeled with $N$-grams and combined with a classifier can predict the accents of the sentence.

Finally, two sentences with the same prominence and prosodic structure can be distinguished by their tune. The *tune* of a sentence is defined as the rise and fall of its fundamental frequency over time. One example could be that the difference between statements and yes-no questions in English is that in yes-no questions there is an rise in the end of the sentence called question rise and in statements there is a falling called final fall. In practice, synthesisers only distinguish between three types of tunes: *continuous rise* (when there are some nouns separated by commas), *question rise* (at the end of yes-no questions) and *final fall* (otherwise).

**Observation.** We only focused in simple models of prosody. Recently, more complex models has appeared and some of them such as ToBI and Tilt models can be found in [28].
3.10. Waveform synthesis

Once we have the phonetic representation of the text, the synthesiser has to create the speech sound. We will explain the *diphone concatenative synthesis*. A diphone is a phonetic unit that goes from the middle of a phoneme to the middle of the next one. The idea of this type of synthesisers is to search in a database all the diphones and concatenate them with some signal processing in the boundaries and changing the prosodic features.

So, once we have the sequence of diphones from the database, we only need to concatenate them and adjust the prosodic features. To concatenate two diphones some aspects must be taken into account. First, we need to apply a windowing function to obtain zero amplitude to the juncture and prevent perceptible click. Moreover, if the diphones are voiced, we need to synchronise the pitch of both diphones. The pitch of a sound is the mental sensation or perceptual correlate of the fundamental frequency.

Now, once the concatenation of the diphones is done, we have to modify the prosodic features of the sentence. There is a simple algorithm to do it called *Time-Domain Pitch-Synchronous OverLap-and-Add (PD PSOLA)*. This algorithm it is explained in different works such as [26, 28].

---

5 To know about other techniques see [26, 28]
Chapter 4
Pattern Theory

4.1. Introduction

In this chapter we deal with pattern theory. Pattern theory tries to analyse all types of signals by creating statistical models to describe them. This method allows to treat in a unified way pattern synthesis and pattern analysis, and pattern analysis includes pattern recognition and pattern classification in the sense of [11, 58, 2].

The models created in pattern theory depend on the data to be studied. The main requirement is that the samples produced by the model (synthesis) must look and feel like the data in the analysed signals. To achieve this may require much work, but once a pattern model is defined, then the synthesis amounts to producing samples or instances of that model, while analysis is mainly concerned with finding out the sample that best fits a piece of data. More generally, since pattern models are abstract structures, they may be transformed by all sorts of algorithms, thus opening a wide field of pattern processing.

Another interesting aspect is that usually pattern models appearing in some kind of signals can be successfully used in other areas. This stems from the fact that problems in quite different fields may have a common underlying structure. An example of this is a HMM model for speech recognition that can also be used in machine translation (cf. 4.5). Actually, in multilingual machine translation there is another pattern approach (cf. [47, 48] for the case of mathematical texts) in which analysis amounts to extracting the semantical content by means of abstract grammar objects (a special case of patterns) and multilingual translation (synthesis) is achieved by rendering that meaning into any language for which a so-called concrete grammar is defined.

The chapter starts by presenting a mathematical approach to patterns adapted to our needs. After introducing the two types of pattern that will be used, we summarize what we need of Bayesian probability theory. As we will see, this frame is enough to show that in pattern theory analysis and synthesis are a common problem, or two sides of the same coin (cf. 4.5).

The main references for this chapter are [21, 38]. For a summary of the theory, see [20], and for an excellent perspective on its aims and significance we refer to the paper [39]. Other classical books on the subject are [17, 18, 19].
4.2. Pattern formalism

In this section we give brief indications about the notion of pattern and the mathematical structures it involves. Since we do not need the most general notion, we take a somewhat restricted variant that is sufficient for our purposes (cf. [38], p. 5):

- To describe patterns present in data, suitable hidden random variables are usually required.
- Observed and hidden variables, and their relevant relations, are typically captured by a graphical model: a graph whose vertices represent the variables and whose edges represent the relations among the variables.
- The models have to allow for pattern synthesis (sampling the model variables) and for pattern analysis (estimating the sample that best fits concrete data).

In general terms, a signal is a function \( f : X \rightarrow V \). Its domain can be continuous (e.g. an speech signal) or discrete (e.g. the nodes of a graph) and the range can be a vector space or some suitable discrete object.

We will distinguish two types of useful patterns concerning \( f \): value patterns and geometrical patterns.

**Value patterns.** These are useful in cases intended to represent the signal as a linear combination of fixed basis functions \( f_k : X \rightarrow V \):

\[
 f \approx \sum_k c_k f_k.
\]

Here the observed data may be a sample vector of \( f \) and the coefficients \( c_k \) are the hidden variables.

Fourier expansions, wavelet expansions, and so on, can be understood as patterns of this type. In such cases, the features of the pattern (including the values of the hidden variables) are computed directly from the values of \( f \) or from some linear combination of them (e.g. power in some frequency band).

**Geometric patterns.** These are meant to cope with non-stationary signals by listing the distinct objects and processes affecting the signal as hidden variables. In language processing, these variables are the different words, noun phrases, verb phrases, sentences and the whole text. An example of a sentence hierarchy is shown in Figure 1. Used for parsing, the observed variable is the text string and the hidden variables are the syntactical generators (see Table 1) placed at the other nodes, while for speech synthesis the hidden variables would be the voice stream corresponding to the text. In vision, the hidden variables are the different parts of the viewed scene, such as objects, people or background. As in the preceding language example, the objects form a hierarchy. This means that the hidden variables can be modeled as a tree of subsets \( X_a \subset X \) such that for every node \( a \) with children \( b \)

\[
 X_a = \bigcup_{a \rightarrow b} X_b.
\]

In our applications in Chapters 5 and 6 we use patterns of each type. In Chapter 6 we distinguish phonemes by their frequency components. In other words, we use value patterns. In this case, the elements of the pattern are the different sinusoids of the Fourier transform and the hidden variables are its coefficients.

In the mosaic application in Chapter 5 the main idea is encoded by the tree in Figure 2. In this case, the nodes contain the elements that enter into the description of the group of symmetries of a mosaic.

---

1 These trees can often be generated by a grammar. These grammars are known as image grammars. We refer to [62] for an introduction to this methodology.
4.3. Bayesian probability

One of the basis of pattern theory is to use the Bayesian probability theory. In a Bayesian approach, we start with learning the models and verifying them explicitly by stochastic sampling, and then we seek algorithms...
that can be applied to practical problems. This fact is an important characteristic of pattern theory: It allows to treat modeling separately from pattern processing.

The problem we wish to solve is to infer the value of hidden random variables $S$ given some observed variables $I$. In other words, we want to calculate $P(S|I)$. For this Bayesian theory is most handy. From the definition of conditional probabilities we have

$$P(S|I)P(I) = P(S, I) = P(I|S)P(S)$$

So, dividing by $P(I)$, we obtain the Bayes’ rule,

$$P(S|I) = \frac{P(I|S)P(S)}{P(I)} = \frac{P(I|S)P(S)}{\sum_{S'} P(I|S')P(S')} \propto P(I|S)P(S)$$

that gives the posterior probability $P(S|I)$ in terms of the prior probability $P(S)$ and the likelihood $P(I|S)$. In other words, Bayes’ rule expresses the probability $P(S|I)$ of the state $S$ given the measurements $I$ in terms of the probability $P(S)$ of the state and the probability $P(I|S)$ of those measurements given $S$.

In Pattern theory, Bayesian models are used because of two main facts. First, we can use it to perform probabilistic inference, such as finding the most probable state of the world given some measurements of a signal. This process is called the maximum a posteriori (MAP) estimate of the world. This type of processes are very common in signal processing.

Secondly, we can generate new samples from the model: assign values to the hidden variables $S$, according to their distribution laws, and output the corresponding $I$. The fitness of the model is judged according to the similarity of the $I$ so generated to the real signals.
4.4. Pattern analysis and Pattern synthesis

An important fact in pattern theory is that from its perspective the analysis of the patterns in a signal and the synthesis of new signals are two sides of the same coin, as they share the probabilistic model.

Pattern theory suggests two convenient views to be taken into account in pattern processing: bottom-up and top-down. They are represented in Figure 3 for a generic pattern. The specializations to mosaic patterns in Chapter 5 and to speech patterns in Chapter 6 are summarized in the caption.

![Fig. 3. Generic stages of pattern theory. For the mosaic patterns in Chapter 5 the model is based on the symmetries of the mosaic and the sampling consists in choosing the type of mosaic and a decoration. For the speech patterns in Chapter 6 the model is based on frequency and the sampling consists in choosing features for a phoneme and also its duration and fundamental frequency.]

Most of the algorithms in pattern recognition only have the bottom-up stage. This stage starts with the signal and calculates a vector of features to characterise the behaviour of the signal. After that, it compares these features with those expected from the different categories. The problem with this type of algorithm is that there is no way to reverse the process, i.e., it is not possible manner to create new samples from the features. Another problem with feature-extracting algorithms is that they are not flexible to cope with anything unexpected, such as a shadow in a text. In contrast, pattern theory can reconstruct the signal, or an approximation of it, and can create endless new samples from the model.

4.5. Machine translation model

To end this chapter we present an example of an approach to machine translation using Bayes’ theorem. We want to translate some sentence from one language to another with a model whose parameters are derived from the analysis of (aligned) bilingual text corpora.

This model is based on two facts. First, a two-language dictionary with frequencies and second, a simple model of short word sequences in the target language and the usage of Bayesian theory to combine them. To
be more specific, we suppose that he have the word string \( f \) in language \( F \) and let \( e \) be a word string in language \( E \). A \( F \)-to-\( E \) dictionary gives a probability distribution of \( P(\mathbf{e} | \mathbf{f}) \), whose higher values yield reasonable translations. If we use the Bayes’ rule, we can use the \( E \)-to-\( F \) dictionary to calculate this probability.

So, we fix \( \mathbf{f} \) and we seek

\[
\mathbf{e} = \arg \max_{\mathbf{e}} P(\mathbf{f} | \mathbf{e}) P(\mathbf{e})
\]

The probability \( P(\mathbf{f} | \mathbf{e}) \) is given by the \( E \)-to-\( F \) dictionary and \( P(\mathbf{e}) \) can be calculated from an \( n \)-gram model (See 3.5). This equation is called the maximum a posteriori (MAP) estimate of \( \mathbf{e} \).

Although a word to word translator sometimes produces non coherent translations, the model proposed here performs a natural translation if we have a reasonable amount of phrases.

The key of this type of machine translation is that one needs a new random variable to describe the relation between \( \mathbf{e} \) and \( \mathbf{f} \): the alignment of the two sentences. In other words, many pairs of words \( e_i, f_j \) will be simple translation one of another, but they may appear in different order. However, there are cases where some words in the language \( F \) get expanded to phrases in language \( E \) (for example: "headache" (English)→ "mal de cap" (Catalan)). Some other cases are that idiomatic phrases can be different or some words have to be interleaved in the other language. Thus, an alignment is a correspondence associating one word \( e_j \) to disjoint strings of \( \mathbf{f} \). So, the dictionary gives a list of strings \( f(1), \ldots, f(l) \) with their probabilities \( P(f(k)|\mathbf{e}) \) as possible translations of the sentence \( \mathbf{e} \). The strings can contain empty strings, singletons and longer strings.

To sum up, the advantages of statistical machine translation are:

- Statistical machine translation systems are not tailored to any specific pair of languages.
- Other translation systems are likely to result in Literal translation. While it appears that statistical systems should avoid this problem and result in natural translations.
- New statistical methods incorporate syntax or quasi-syntax structures that improve the translation process.
Chapter 5
Wallpaper group algorithm

This chapter is devoted to present an algorithm for the synthesis and classification of plane mosaics based on their symmetry groups. The algorithm is based on the mathematical classification of these symmetry groups (see Chapter 2) and it is meant to be a showcase of the Grenander-Mumford approach to patterns (see Chapter 4). The algorithm is implemented in MATLAB and its functionality in regard to the generation of mosaics and their classification is illustrated with examples, including mosaics generated by the program.

5.1. Mosaic generator

In this section we present a MATLAB program which implements an algorithm that yields random mosaics with a random decoration.

The algorithm can be divided into four steps:

1. To create an image that will be used for the decoration of the mosaic.
2. To chose at random a mosaic type and to build its standard cell with that decoration.
3. To tile the plane with that cell.
4. To transform the mosaic by randomizing its parameters according to the degrees of freedom of its type.

5.2. Decoration generator

In this part of the algorithm we explain our method for creating an image to decorate a mosaic tile. As the decoration of a mosaic can be an arbitrary image, the first decision was to avoid this dull generality and to device instead an interesting procedure inspired in pattern theory.

To that end, the possibility that seemed more appealing was to mimic the style traits of a painter and use them, following the tenets of pattern theory, as rules to generate random decorations. Since that would be rather laborious, and to a large extend peripheral to our concerns here, we have followed instead a simpler approach that is sufficient to illustrate the idea.

Our decorations are formed by picking a colour for the background and then inserting circles at random (their number, centers, radii and colours are picked at random). The only restriction that we impose is that...
the circles are not meet the border or the diagonals of the tile. This restrictions are only meant to avoid discontinuities (usually they are not very pleasing) when crossing tile boundaries. See Figure 1 for output samples of this procedure.

5.3. Main cell creation

This part of the algorithm takes a decoration image and a type of wallpaper group and returns the main cell of the mosaic.

The algorithm first decides which part of the image is taken according to the mosaic type. If the mosaic has a quadrilateral fundamental domain, the program applies an affine transform to the image to make it the fundamental domain. If the mosaic has a triangular fundamental domain, the program only uses a half of the image.

Once the fundamental domain is obtained, the algorithm creates the main cell by applying the symmetries of the wallpaper group to the fundamental domain. Finally, the main cell is transformed into a square cell to ease the task of next step, which actually produces the tiling.

Figure 2 displays some examples.
5.4. Tiling

This application creates the tiling of the input image. First, the algorithm creates a bigger image formed by a tiling of the original. After that, the program applies a affine transform to the new image depending of the degrees of freedom of the lattice. These degrees of freedom can be the length of the main cell sides and the angle between them.

The degrees of the lattice are listed in the table 1.

<table>
<thead>
<tr>
<th>Lattice type</th>
<th>Degrees of freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallelogrammatic</td>
<td>3 degrees: The length of the two sides and the angle.</td>
</tr>
<tr>
<td>Rectangular</td>
<td>2 degrees: The length of the two sides.</td>
</tr>
<tr>
<td>Rhombic</td>
<td>2 degrees: The length of the sides and the angle.</td>
</tr>
<tr>
<td>Square</td>
<td>1 degree: The length of the sides.</td>
</tr>
<tr>
<td>Hexagonal</td>
<td>1 degree: The length of the sides.</td>
</tr>
</tbody>
</table>

Table 1. The degrees of freedom of the different type of lattices.

Mosaics of every type created by the program are listed in 5.9.

5.5. Classification algorithm

In this section we present the classification algorithm. In some sense it makes the opposite of the previous program.

The application assumes that a mosaic image is given as a parallel projection in the direction of its perpendicular lines, i.e., from a frontal perspective (see Figure 3), and returns the type of wallpaper group of the mosaic.

Our computational algorithm for periodic pattern perception based on wallpaper groups includes two main components:

1. recovering the underlying translation lattice of the pattern.
2. classifying the wallpaper group of the image.

The key tool to detect periodic patterns is the correlation function. We use it as a probability function, since we find in every type of isometry the maximum value of the correlation function between the image and the transformation of the image due to the isometry. To make it a probability function we divide it by the maximum value of the autocorrelation of the original image.

Let $I(x, y)$ be an image and let $\varphi$ be an isometry. To calculate the probability that the image $I$ has this isometry we use the following steps.

1. compute the transformed image $J(x, y) = \varphi(I(x, y))$.
2. calculate the median of each image.
5. WALLPAPER GROUP ALGORITHM

Fig. 3. Image perspective.

(3) subtract the median from both images and obtain \( \hat{I}(x,y) \) and \( \hat{J}(x,y) \).
(4) compute the correlation function using the FFT by \( r_{ij}(x,y) = \mathcal{F}^{-1}\{\mathcal{F}^*(-\hat{I}(x,y))\mathcal{F}\{\hat{J}(x,y)\}\} \)
(5) calculate the maximum of the autocorrelation function of \( \hat{I}(x,y) \) as \( M_{ij} = r_{ij}(x,y) = r_{ij}(0,0) \)
(6) divide the correlation function by the previous number, i.e, \( C_{ij}(x,y) = \frac{r_{ij(x,y)}}{M_{ij}} \)
(7) search the maximum of the previous function.

In the next two sections we explain the two parts of our algorithm in more details.

5.6. Translational lattice detection

This part of the algorithm is centered in the processing of the image and the detection of the lattice (this is the most important part).

The image processing starts by transforming the image into a grey scale image, as it is easier to manipulate than a colour image. It is, in a sense, a compression technique. After that, the gray image is compressed by a 1-level wavelet transform using Haar wavelets. The advantage of this compression is to make uniform the different regions of the mosaic and emphasize the boundaries.

Once the image has been processed, the algorithm tries to find the lattice of the wallpaper group. We divide this search in three parts:

(1) Finding the peaks of the autocorrelation function of the image
(2) Finding the region of dominance of every peak
(3) Calculating the main vectors of the lattice.

5.6.1. Peak finding. The first step to determinate the lattice of the wallpaper group is to find the peak of the autocorrelation function. To do this in our case we first decompose the autocorrelation image in squares of \( n \) (typically 5) pixels length per side. After that, we calculate the maximum value of every square and keep the position of the peak inside the square.

Once the maximum for each square has been found, we find the maximum of them and save the value and the positions where it occurs. With this method we avoid some noise in the autocorrelation function produced by noise in the original image.
5.6.2. Region of dominance. It is a nontrivial task to find a proper set of peaks in an autocorrelation surface of a periodic pattern. After some experimental trials, our observation is that the absolute height of a peak is not as important as the size of its region of dominance, defined as the largest circle centered on the candidate peak such that no higher peaks are contained in the circle. A peak with a low height, but located far from any larger neighbours, perceptually is much more important than a high peak that is close to an even higher one.

So, given the peak candidate for the previous part of the algorithm, we order them according to their region of dominance and keep only a percentage of them.

5.6.3. Determine the lattice vectors. Having extracted a set of candidate lattice points as dominant peaks in the autocorrelation image, the next task is to find the shortest linearly independent translation vectors that generate the lattice.

To determine the lattice vectors, we first find the peak nearer to the center of the autocorrelation function. Since this is always a peak we use it to find the vectors. After that, we calculate the 2 nearer peak such that the vectors are linearly independent. To avoid bad peaks, we check that at double size there is another peak.

Once the lattice has been found, we determine the lattice type by calculating the angle between the vectors and the ratio between their lengths.

5.7. Classifying the wallpaper groups

This part of the algorithm calculates the probabilities of the main cell to have the different types of symmetries. After that, the wallpaper group inside the image is determined using these probabilities.

Once the lattice of the wallpaper group has been found, the first step to calculate the wallpaper group of the image is to apply an affine transform to the image to obtain an square main cell. A square main cell makes easier to calculate the value of correlations inside the image.

Although it would seem that in this form it would be more difficult to calculate symmetries due to the composition of the symmetries with the affine transform, almost every symmetry, except the rotations of order 3 and 6, has diagonal matrix and commutes with the affine matrix transform. So, in these two cases one has to be very careful to compute in the right order the matrix transformations.

The algorithm used to determine the wallpaper group is described in the following lines. The input of the algorithm is the lattice type computed (given by translation vectors $T_1$ and $T_2$) by measuring the angles between them, their lengths and an image of the main cell. In the following text we will use the notation $T_1$-reflection ($T_2$-reflection) referred to a reflection about an axis of the unit lattice parallel with $T_1$ (respectively, $T_2$).

Euclidean Algorithm:

1. If the lattice is a parallelogram lattice (two possible groups: $p1$, $p2$), test if there is a 2-fold rotation. If the answer is yes, $W$ is a $p2$, otherwise it is $p1$.
2. If the lattice is a rectangle lattice, there are five plus two (seven) possible groups: $pm$, $pg$, $pmm$, $pmm$, $pgg$ and $p1$, $p2$. First test if there is a 2-fold rotation. If it exists (four possible groups: $pmm$, $pgm$, $pgg$,
5.8. Examples of the algorithm

The examples in this section illustrate the symmetry classification algorithm. We present 3 different examples. The first two examples are computer generated mosaics and hence they have few imperfections and their symmetries are very clear. The last example is an Indian metalwork presented at the Great Exhibition in 1851. Since it is human made, it has some imperfections and those affect significantly the correlation values.

Figure 4 displays the mosaic chosen for the first example. The wallpaper group is a $pmg$ and we will use our program to identify it.

First of all, the program transforms the image into a grey scale one and compresses it with a 1-level wavelet transform. After that, it computes the autocorrelation function of the image and finds the peaks of the autocorrelation with the algorithm described in Figure 5.6. Figure 5 shows the peaks found by the program.

After that, we determine the main cell of the lattice. This can be seen in Figure 6. In Figure 7, the main cell with the 2-fold rotation point in the center is shown.

Once the lattice of the wallpaper group is found, and since the lattice type is a rectangular lattice, we test for a 2-fold rotation center. The program returns a value of $P(2\text{-fold}) = 0.9276$. Table 2 contains the probability values of all the symmetries. So, following the algorithm described in 5.7, the only possible groups are $p2$, $pmg$, $pmm$ and $pgg$. Now we test for $T_1$ and $T_2$ reflections. The result can be seen in Table 2. So, as there is one reflection and one glide reflection, the wallpaper group is a $pmg$.
5.8. EXAMPLES OF THE ALGORITHM

Fig. 4. A computer generated $pmg$ wallpaper group.

Fig. 5. A computer generated $pmg$ wallpaper group.

Fig. 6. The autocorrelation function of Figure 4 and the peaks found by the algorithm.

Fig. 7. The main cell of the mosaic in Figure 4 with a 2-fold rotation point in the center.

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.9276</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.1972</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.2205</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.2011</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.9720 with a glide of 25</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.8841 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.2221 with a glide of 32</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.2129 with a glide of 25</td>
</tr>
</tbody>
</table>

Table 2. Probabilities of the different symmetries for Figure 4. The size of the images after the affine transform is $50 \times 50$.

Figure 8 shows the mosaic chosen for the second example. The wallpaper group is a $p6m$, which is the wallpaper group with more symmetries, and we will use our program to identify it. We notice that the image has some imperfections and due to that the probabilities are lower than the previous example. However, the good values are still higher than 0.5.
As in the previous example, the program transforms the image into a grey scale image and compresses the image with a 1-level wavelet transform. After that, it computes the autocorrelation function of the image and finds the peaks of the autocorrelation with the algorithm described in 5.6. Figure 9 shows the peaks found by the program.

After that, we determine the main cell of the lattice. This can be seen in Figure 10. Figure 11 shows the main cell after the affine transformation.

Once the lattice of the wallpaper group is found and since the lattice type is an hexagonal lattice, we search for 3 rotation centers. As the program finds that the image has 3 rotation center with a probability of 0.7133, we conclude that the image has 3-fold rotations. Then we test the 2 rotation centers and we find one with a probability of 0.7284. So, the wallpaper group can only be \( p6 \) or \( p6m \). Finally, we test for some symmetries and we find them. Thus, the group is a \( p6m \). Table 3 lists all the probabilities.

Figure 12 shows the mosaic chosen for the last example. This mosaic is human made and due to that fact the probabilities found by the algorithm are lower, although they are still higher than 0.5. The wallpaper group is a \( cm \).
5.8. EXAMPLES OF THE ALGORITHM

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.7284</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.7133</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.2749</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.6455</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.9123 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.8970 with a glide of 25</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.9495 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.8867 with a glide of 25</td>
</tr>
</tbody>
</table>

Table 3. Probabilities of the different symmetries of Figure 8.

As in the previous examples, the program transforms the image into a grey scale one and compresses the image with a 1-level wavelet transform. As this image is human made, the compression has a higher effect in it. Although it is very difficult to see with a human eye, the program obtains some improved probabilities for the compressed image. After that, it computes the autocorrelation function of the image and finds its peaks. Figure 14 shows the peaks found by the program.

Fig. 12. Indian metalwork at the Great Exhibition in 1851.

Fig. 13. The lattice of Figure 12

Fig. 14. The autocorrelation function of Figure 12 and its peaks.

Fig. 15. The main cell of the mosaic in Figure 12 after the affine transformation.

After that, we determine the main cell of the lattice. This can be seen in Figure 15. Figure 15 shows the main cell after the affine transformation.
Once the lattice of the wallpaper group is found, and since the lattice type is a rhombic lattice, we search for diagonal rotation centers. As the program returns a value of 0.7518, the algorithm concludes that the wallpaper group can only be \( cm \) or \( cmm \). Finally, the program searches for a 2 rotation center and as the returned value is 0.3495, it infers that the wallpaper group of the image is \( cm \). Table 4 shows all the probabilities.

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.3495</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.1865</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.2243</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.2365</td>
</tr>
<tr>
<td>( T_1 ) reflection</td>
<td>0.2182 with a glide of 0</td>
</tr>
<tr>
<td>( T_2 ) reflection</td>
<td>0.2335 with a glide of 0</td>
</tr>
<tr>
<td>( D_1 ) reflection</td>
<td>0.3548 with a glide of 25</td>
</tr>
<tr>
<td>( D_2 ) reflection</td>
<td>0.7518 with a glide of 25</td>
</tr>
</tbody>
</table>

Table 4. Probabilities of the different symmetries of Figure 12

5.9. Classification of generated mosaics

In this section we show the values of the different symmetries found by the classification program for mosaics synthesized by the creation algorithm. The values are shown below.

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.4231</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.0713</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.0948</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.1459</td>
</tr>
<tr>
<td>( T_1 ) reflection</td>
<td>0.3531 with a glide of 0</td>
</tr>
<tr>
<td>( T_2 ) reflection</td>
<td>0.1237 with a glide of 1</td>
</tr>
<tr>
<td>( D_1 ) reflection</td>
<td>0.3136 with a glide of 0</td>
</tr>
<tr>
<td>( D_2 ) reflection</td>
<td>0.3300 with a glide of 26</td>
</tr>
</tbody>
</table>

Table 5. Symmetry probabilities of \( p1 \)
5.9. CLASSIFICATION OF GENERATED MOSAICS

Table 6. Symmetry probabilities of $p_2$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.9278</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.0889</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.0996</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.0986</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.1579 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.1546 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.2285 with a glide of 18</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.3984 with a glide of 24</td>
</tr>
</tbody>
</table>

Table 7. Symmetry probabilities of $pm$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.3341</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.1798</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.2005</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.1794</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.8905 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.3117 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.2547 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.2278 with a glide of 10</td>
</tr>
</tbody>
</table>

Table 8. Symmetry probabilities of $pg$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.3743</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.2633</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.2872</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.0922</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.8822 with a glide of 25</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.3184 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.3083 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.2666 with a glide of 49</td>
</tr>
</tbody>
</table>

Fig. 17. Image of a generated $p_2$

Fig. 18. Image of a generated $pm$

Fig. 19. Image of a generated $pg$
Fig. 20. Image of a generated $cm$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.3868</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.3087</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.3570</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.3687</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.3563 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.3093 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.3945 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.8052 with a glide of 25</td>
</tr>
</tbody>
</table>

Table 9. Symmetry probabilities of $cm$

Fig. 21. Image of a generated $pmm$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.9051</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.2429</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.3912</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.0684</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.8644 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.9181 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.3716 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.2541 with a glide of 9</td>
</tr>
</tbody>
</table>

Table 10. Symmetry probabilities of $pmm$
5.9. CLASSIFICATION OF GENERATED Mosaics

Fig. 22. Image of a generated \( pmg \)

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.8268</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.1914</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.3263</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.1262</td>
</tr>
<tr>
<td>( T_1 ) reflection</td>
<td>0.8202 with a glide of 0</td>
</tr>
<tr>
<td>( T_2 ) reflection</td>
<td>0.8340 with a glide of 25</td>
</tr>
<tr>
<td>( D_1 ) reflection</td>
<td>0.3148 with a glide of 0</td>
</tr>
<tr>
<td>( D_2 ) reflection</td>
<td>0.2440 with a glide of 9</td>
</tr>
</tbody>
</table>

Table 11. Symmetry probabilities of \( pmg \)

Fig. 23. Image of a generated \( pgg \)

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.8757</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.1773</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.2043</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.1298</td>
</tr>
<tr>
<td>( T_1 ) reflection</td>
<td>0.8699 with a glide of 25</td>
</tr>
<tr>
<td>( T_2 ) reflection</td>
<td>0.8283 with a glide of 25</td>
</tr>
<tr>
<td>( D_1 ) reflection</td>
<td>0.3408 with a glide of 0</td>
</tr>
<tr>
<td>( D_2 ) reflection</td>
<td>0.1669 with a glide of 43</td>
</tr>
</tbody>
</table>

Table 12. Symmetry probabilities of \( pgg \)
Table 13. Symmetry probabilities of \textit{cmm}

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.9204</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.1159</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.0868</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.2149</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.3639 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.0968 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.9026 with a glide of 25</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.8339 with a glide of 0</td>
</tr>
</tbody>
</table>

Table 14. Symmetry probabilities of \textit{p4}

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.8798</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.1750</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.9202</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.1991</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.1142 with a glide of 42</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.1113 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.3010 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.0857 with a glide of 45</td>
</tr>
</tbody>
</table>

Table 15. Symmetry probabilities of \textit{p4m}

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.9337</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.2060</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.9446</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.2219</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.9516 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.9463 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.8963 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.9067 with a glide of 0</td>
</tr>
</tbody>
</table>

Fig. 24. Image of a generated \textit{cmm}

Fig. 25. Image of a generated \textit{p4}

Fig. 26. Image of a generated \textit{p4m}
5.9. CLASSIFICATION OF GENERATED MOSAICS

Fig. 27. Image of a generated \( p4g \)

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.9366</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.0723</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.8430</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.2744</td>
</tr>
<tr>
<td>( T_1 ) reflection</td>
<td>0.8867 with a glide of 25</td>
</tr>
<tr>
<td>( T_2 ) reflection</td>
<td>0.8402 with a glide of 25</td>
</tr>
<tr>
<td>( D_1 ) reflection</td>
<td>0.7884 with a glide of 0</td>
</tr>
<tr>
<td>( D_2 ) reflection</td>
<td>0.7669 with a glide of 0</td>
</tr>
</tbody>
</table>

Table 16. Symmetry probabilities of \( p4g \)

Fig. 28. Image of a generated \( p3 \)

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.3480</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.8444</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.1033</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.3187</td>
</tr>
<tr>
<td>( T_1 ) reflection</td>
<td>0.3831 with a glide of 0</td>
</tr>
<tr>
<td>( T_2 ) reflection</td>
<td>0.1559 with a glide of 0</td>
</tr>
<tr>
<td>( D_1 ) reflection</td>
<td>0.2099 with a glide of 0</td>
</tr>
<tr>
<td>( D_2 ) reflection</td>
<td>0.2181 with a glide of 24</td>
</tr>
</tbody>
</table>

Table 17. Symmetry probabilities of \( p3 \)
### Table 18. Symmetry probabilities of $p3m1$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.2283</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.8744</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.2693</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.2208</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.2765 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.2757 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.9554 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.3035 with a glide of 27</td>
</tr>
</tbody>
</table>

### Table 19. Symmetry probabilities of $p31m$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.2623</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.8238</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.3869</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.0315</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.7992 with a glide of 0</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.8139 with a glide of 0</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.3903 with a glide of 27</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.8067 with a glide of 0</td>
</tr>
</tbody>
</table>

--

**Fig. 29.** Image of a generated $p3m1$

**Fig. 30.** Image of a generated $p31m$
5.9. CLASSIFICATION OF GENERATED MOSAICS

Fig. 31. Image of a generated $p6$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.8655</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.8671</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.1633</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.8505</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.1517 with a glide of 23</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.1331 with a glide of 38</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.3075 with a glide of 0</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.3320 with a glide of 24</td>
</tr>
</tbody>
</table>

Table 20. Symmetry probabilities of $p6$

Fig. 32. Image of a generated $p6m$

<table>
<thead>
<tr>
<th>Symmetry</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 rotation</td>
<td>0.8160</td>
</tr>
<tr>
<td>3 rotation</td>
<td>0.9141</td>
</tr>
<tr>
<td>4 rotation</td>
<td>0.1446</td>
</tr>
<tr>
<td>6 rotation</td>
<td>0.9177</td>
</tr>
<tr>
<td>$T_1$ reflection</td>
<td>0.9024 with a glide of 25</td>
</tr>
<tr>
<td>$T_2$ reflection</td>
<td>0.8431 with a glide of 25</td>
</tr>
<tr>
<td>$D_1$ reflection</td>
<td>0.9694 with a glide of 25</td>
</tr>
<tr>
<td>$D_2$ reflection</td>
<td>0.8643 with a glide of 25</td>
</tr>
</tbody>
</table>

Table 21. Symmetry probabilities of $p6m$
5.10. MATLAB functions

In this section we present the principal functions coded in MATLAB to implement the algorithms described before.

5.10.1. Generator algorithm. First, we present the functions defined to generate a mosaic. These functions are listed below.

- function \([I]=\text{imagegenerator}\)
  
  - \(I\): A square image used as decoration of the mosaic.
  
  - This function creates a random decoration for the new mosaic. To do it, it applies the algorithm described in 5.2 and it uses the MATLAB function \(\text{random}\) to decide the colours, the number of circles, and their attributes.

- function \([C,T]=\text{cellcreation}(i)\)

  - \(i\): Image of the decoration.
  
  - \(C\): A square main cell of the mosaic.
  
  - \(T\): Type of mosaic.
  
  - This function creates a square version of the main cell of the mosaic as described in 5.3. In the case of mosaics with rotations of order 3, it uses the function \(\text{imtransform}\). To decide the mosaic type, it uses the function \(\text{random}\).

- function \([M]=\text{tiling}(c,t)\)

  - \(c\): Image of a square main cell of the mosaic.
  
  - \(t\): Type of mosaic.
  
  - \(M\): Image of the whole mosaic.
  
  - This function creates a tiling with the square main cell and then applies an affine transform to the tiling to create the mosaic with the MATLAB function \(\text{imtransform}\). To define the affine matrix it applies a \(\text{random}\) to every degree of freedom of the mosaic lattice.

5.10.2. Classification algorithm. In this section we present the main functions, again coded in MATLAB, to classify the different mosaics. For every function we present the inputs (in lower-case letters), outputs (in capital letters) and a little explanation of what the function does.

- function \([X,A]=\text{mosaics}(i)\)

  - \(i\): The image of the mosaic.
  
  - \(X\): The image in grey scale.
  
  - \(A\): The autocorrelation of the image.
  
  - This function transform the original image in a gray scale and calculates its autocorrelation function. It uses the \(\text{fft}\) function of MATLAB.

- function \([P,V,R]=\text{maxima}(a,x)\)

  - \(a\): Autocorrelation function.
  
  - \(x\): The image in gray scale.
  
  - \(P\): Peaks of the autocorrelation function.
  
  - \(V\): Vectors of the lattice.
  
  - \(R\): Type of lattice.
  
  - The purpose of this function is to calculate the peaks of the autocorrelation and the lattice of the mosaic from the image and its autocorrelation. It is uses the algorithm described in 5.6.1.
• function [A,J]=squarecell(i,v)
  – i = image of the mosaic.
  – v = vectors of the lattice.
  – A = matrix of the affine transformation.
  – J = square main cell of the mosaic.
  – This function takes the image of the mosaic and the lattice and returns an square main cell of the mosaic. It applies an affine transform with the Matlab function `imtransform`.

• function [J,V]=rotation(i,t)
  – i = image of the mosaic.
  – t = type of rotation.
  – J = image after the rotation.
  – V = maximum value of the correlation between the image and the rotation of it.
  – This part of the algorithm calculates the probability that there is a rotation of type t. It applies a transformation to the image with the Matlab function `imtransform` and calculates the maximum of the correlation of the original image and the transformed one.

• function [V,G]=reflection(i,t)
  – i = image of the mosaic.
  – t = type of reflection.
  – V = maximum value of the correlation between the image and the reflection of it.
  – G = value of the sliding offset of the symmetry.
  – This function also applies a transform with the Matlab function `imtransform` and calculates the maximum in the direction of the reflection in the correlation image.
Chapter 6
Speech classification

As we have indicated in Chapter 4, in this chapter we show a MATLAB application that parses the phonemes of a given recorded voice and another one that generates new recorded voices.

The first sections of this chapter deal with the parsing of words into phonemes. We use the DWT to parse a word into its phonemes because it has the property to extract the power of the signal in different subbands of frequencies. The analysis of the power in different frequency bands has the capability to distinguishing the beginning and ending of the phonemes because the power in the different subbands is roughly constant over the lifetime of the phoneme [13, 63, 64].

The second part deals with the classification of the phonemes. We use an artificial neural network (see Appendix A) trained from the first formants of the different phonemes (Chapter 3). The formants are features that can discriminate phonemes (especially voiced phonemes).

Finally, the last sections of the chapter are devoted to explains a method to synthesise voices from formants. Using the formants from the database we have obtained an algorithm that creates samples that are different from those in the initial database (artificial voices).

6.1. Classification algorithm

The application is a program which returns, given a recording of a person saying a word, its sequence of phonemes. In our case, we use a data base of some Catalan words spoken by children.

The algorithm consist in two main components:

1. Segmentation: it divides the spoken word into phonemes.
2. Identification: classifies each phoneme using a neural network.

The first component uses the properties of the local spectrum of the wavelet transform to find the points where the energy in one subband changes, and this determines the phoneme boundaries. The second component calculates the formants of each phoneme and uses them as features for classification.

In the next two sections we explain the two parts of our algorithm in more detail.
6.2. Speech segmentation

6.2.1. Segmentation algorithm. This part of the algorithm divides the spoken word into phonemes by detecting their boundaries using the DWT. The DWT is used because of its decomposition in frequency subbands. The analysis of the power in different frequency subbands is an excellent way to distinguish the beginning and the end of phonemes, as many phonemes exhibit rapid changes in particular subbands, and these provide evidence for their initial and end points. The start of a phoneme should be marked by an initially small but rapidly rising power level in one or more of the DWT levels. In other words, we should expect the power to be small and the derivative to be large.

Some experiments as in [63] showed that the speech signal should be decomposed into six levels, which cover the frequency band of a human voice. In our case we use a 6-level Symlets DWT.

Let s be the speech signal. First of all we calculate its M-DWT as \( DWT(s) = \{d_M, d_{M-1}, \ldots, d_1, a_1\} \). As the human voice has negligible power at very low frequencies, we only will use the \( d_j \) coefficients.

After that as the wavelet spectrum samples in \( n \)-level depends on the length \( N \) of speech signal in time domain according to \( 2^{-M+n-1}N \), where \( n = 1, \ldots, M \), the power at each band is calculated in a different way to obtain the same number of samples following the equation

\[
p_n(i) = \sum_{j=1}^{2^{n-1}} d_n(j + 2^{n-1}i), \quad \text{where} \quad i = 0, \ldots, 2^{-M}N - 1 \quad (\ast)
\]

The DWT subband power shows rapid variations. So, the first order differences in the power are inevitably noisy. Due to that we calculate the envelopes \( e_{p_n} \) of each subband power function by choosing the highest values of \( p_n \) in a window of given size \( W \) to obtain a power envelope. See table 1.

<table>
<thead>
<tr>
<th>DWT Level</th>
<th>Number of samples in comparison with level 1</th>
<th>Window size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 1. Windows size of each DWT level.

To calculate the derivative of \( p_n \) we use a smoothed differencing operator. The subband power \( p_n \) is convolved with the mask \([1, 2, -2, -1]\) to obtain smoothed rate-of-change information \( r_n \).

After that, we can detect phoneme boundaries searching for \( i \)-points for which the inequality

\[
p \geq || \beta ||r_n(i)|| - e_{p_n}(i)||
\]

holds, where the constant \( p \) is threshold value which accounts for the time scale and sensitivity of the crossing points. In practice we take \( p = 0.02 \). In the inequality, \( r_n \) is multiplied by a scaling factor \( \beta \) which is approximately equal to 1.
The algorithm is summarized in the following steps:

1. Normalise a speech signal by dividing by its maximum value.
2. Decompose a signal into 6-levels of the DWT.
3. Calculate the sum of power samples in all frequency subbands to obtain the $n$-th subband power representation $p_n(i)$ according to ($\ast$).
4. Obtain the envelopes $e_{p_n}(i)$ for functions in each subband by choosing the highest value of $p_n$ in a given window $W$ (see table 1).
5. Calculate the rate-of-change function $r_n(i)$ by filtering $p_n(i)$ with the mask $[1, 2, -2, -1]$.
6. Given a threshold $p$ of the distance between $r_n(i)$ and $e_{p_n}(i)$ and a threshold $p_{\text{min}}$ of minimal $e_{p_n}$, find indexes for which

\[
\left(\left|\left| r_n(i) \right| - e_{p_n}(i) \right| < p \right) \land \left(\left|\left| r_n(i+1) \right| - e_{p_n}(i+1) \right| > p \right) \lor \left(\left|\left| r_n(i-1) \right| - e_{p_n}(i-1) \right| > p \right) \land \left( e_{p_n}(i) > p_{\text{min}} \right).
\]

Write such indices in one vector.
7. Find and group indices where there is no space between neighbouring ones longer than attribute $\Delta$. This attribute is the minimum length of a phoneme which approximately is 0.028 seconds.
8. Calculate an average index value for each group found in the previous step as points of phoneme change.

6.2.2. Examples. The examples in this section serve to illustrate the segmentation algorithm. We present two examples. In the first example, the algorithm divides exactly the speech signal while in the second the program divides the speech signal in more segments than it should and some points are a bit different than the real ones.

Figure 1 shows the recorded waveform of a child saying the word ONA. In figures 3 we can see the power function, its envelope and the rate-of-change function. We can also see the points of phoneme transition at each subband before the average step. Finally, Figure 2 shows the recorded waveform with some red lines that indicate the segmentation points found by the algorithm.

![Fig. 1. The speech waveform of a child saying the word ONA.](image1.png)

![Fig. 2. As in Figure 1 but with the segmentation found by the program (red lines).](image2.png)

The second example of this section is shown in Figure 4. In this example the word is ENA. In Figure 5 the red lines represent the segmentation found by the algorithm and the green lines belong to the true segmentation.

![Fig. 4. The speech waveform of a child saying the word ENA.](image4.png)

![Fig. 5. The speech waveform of a child saying the word ENA with the true segmentation (green lines) and the segmentation found by the algorithm (red lines).](image5.png)
Fig. 3. In cyan, the power function $p_n$ of the $n$-th frequency subband. In black, its envelope function $e_{p_n}$, and in blue, its rate-of-change function $r_n$. The asterisks in red are the points where a phoneme change is likely.
6.3. Phoneme recognition

This second part of the algorithm takes each segment detected by the first part and decides which phoneme it is.

To explain how the program works we have divided it in two steps:

- To determine the characteristics (features) that represent each phoneme.
- To use those characteristics to classify the phoneme.

6.3.1. Feature extraction. Once the speech signal has been segmented, we only keep the corresponding 40 ms samples in the middle of the phoneme. This method is used because the segmentation program often returns segmentation points that are a bit different from the real one. Furthermore, at phoneme boundaries the signal is not stationary and hence they are not useful for classification.

We use the first 5 formants of each phoneme as features. The position of the formants can identify different phonetic sounds. To calculate the formants, we divide the phoneme in segments of 20 ms and calculate the formants of each segment. To calculate the formants of each, we first calculate the LPC (Linear Predictive Coding) coefficients $a_0, \ldots, a_n$ of each segment and then we calculate the roots of the predictive polynomial $A(z) = a_0 + a_1z + \ldots + a_nz^n = 0$. After that, we put all the formants of all segments together and put them in ascendent order. Then, we separate them in groups where two are in a same group if they differ in less than 200 Hz. After that, we calculate the mean of each group and we use the first 5 formants as features to be used in the second step.

6.3.2. Classification. The features extracted in the first step are fed to a MLP (multilayer perceptron). A MLP (cf. Appendix A) consists of one or more hidden layers and an output layer. Usually a single hidden layer is used for classification purposes because it has less computational complexity and better stability during the training phase. However, for highly non-linear classification problems, more hidden layers may be used. This also helps in reducing the problem of over-fitting that arises in MLP when the number of...
neurons in the hidden layer is large. MLP is trained under supervision and has an ability to form a non-linear decision boundary to classify patterns.

### 6.3.3. Experimental results

We have used a simple MLP classifier with two hidden layers, with 10 and 6 neurons, and an output layer. The number of nodes in the output layer is chosen to be equal to the number of classes (six).

To train the MLP, we used a database of spoken words by Catalan children. The phonemes used in this project are listed in Table 2. A total of 100 samples of each phoneme were used. Of those, 90 where used for training the classifier and 20 for testing it.

<table>
<thead>
<tr>
<th>Vowels</th>
<th>/o/</th>
<th>/e/</th>
<th>/i/</th>
<th>/o/</th>
<th>/m/</th>
<th>/n/</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consonants</td>
<td>/m/</td>
<td>/n/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. List of the phonemes extracted from the data base.

The recognition performance achieved by using the energy coefficients per sample is found to be quite high as one can see in Table 3. The values in the diagonal should be 1 if the classifier worked perfectly. The worst value in the diagonal is 0.7 for the phoneme /i/ and the best performance is the 0.9 for the phoneme /e/. On the average, the value of correct classification is 0.82.

<table>
<thead>
<tr>
<th>Proposed feature confusion matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>/o/ 0.87 0.13 0 0 0 0 /e/ 0 0.9 0.1 0 0 0 /i/ 0.08 0.22 0.7 0 0 0 /o/ 0.08 0.08 0 0.84 0 0 /m/ 0.1 0.05 0 0 0.85 0 /n/ 0 0 0.07 0.15 0.07 0.71</td>
</tr>
</tbody>
</table>

Table 3. Confusion matrix obtained with the testing samples.

### 6.4. Examples of the classification algorithm

In this section we present two examples of the whole algorithm.

The first one corresponds to the word OM. In this case the segmentation process returns the correct points of the phoneme boundaries, as one can see in the figure 6. After simulating the MLP, the result is the phonetic transcription /o//m/.

The second example corresponds to the catalan word ONA. As one can see in figure 7, the segmentation algorithm returns more segmentation points and some of them are displayed with respect to the true separations. However, the classifier returns the phonetic transcription /o//o//n//a/.
6.5. Synthesis algorithm

In this section we present the synthesis algorithm that creates a sound from a sequence of five formants and a fundamental frequency.

To synthesise the signal, we first create the spectrum of the signal from the five formants. To this end, we create a sequence of ones of length the ratio between the frequency rate and the fundamental frequency. After that, we pass the sequence to a series of filters that try to represent the glottal source function. Then, for each formant we add a filter that represents the contribution of the formant to the spectrum. The filter equations are

\[ c = -e^{-2\pi W f_R} \]
\[ b = 2e^{-\pi W f_R} \cos(2\pi W f_R)4 \]
\[ a = 1 - b - c \]

\[ H(z) = \frac{a}{1 - b z^{-1} - c z^{-2}} \]

where \( W \) is the bandwidth of the formant and \( f_R \) is the frequency rate. The bandwidth of the different formants can be found in Table 4. The MATLAB code of this part, due to Peter Assmann, is listed in B (function get_vspect). Once the spectrum has been created, we only have to pass it to the time domain for an appropriate duration. Figures 8-13 show some examples of synthesised phonemes by this procedure.

<table>
<thead>
<tr>
<th>Formant</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>90</td>
<td>110</td>
<td>140</td>
<td>180</td>
<td>220</td>
</tr>
</tbody>
</table>

Table 4. Bandwidth of the different formants.
6. SPEECH CLASSIFICATION

(a) Spectrum of the generated phoneme.
(b) Spectrum of a database sample
(c) Time domain plot of the generated phoneme.
(d) Time domain plot of a database sample

Fig. 8. Figures of the phoneme /a/.

6.6. Phoneme generator

In this section we present an algorithm to create new voices (artificial speakers). For this purpose, we need to find an “ideal” representation of each phoneme in the frequency domain. In other words, we want to describe each phoneme as a pattern, and this can be done by using the first formants.

First, we calculate the first 5 formants of each phoneme. After that, for each formant we take all the values and we classify the values of this formant into classes. We take the class with more elements and calculate the mean and standard deviation of this class and take it as the value for the formant pattern of the phoneme.

In table 5 we show the values of the mean and variance of each formant for the six phonemes.

<table>
<thead>
<tr>
<th>Formant</th>
<th>/a/</th>
<th>/e/</th>
<th>/i/</th>
<th>/o/</th>
<th>/m/</th>
<th>/n/</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>563.5 (93.3)</td>
<td>495.4 (97.4)</td>
<td>543.5 (77.5)</td>
<td>452.9 (67.9)</td>
<td>392.7 (73.8)</td>
<td>364.5 (72.5)</td>
</tr>
<tr>
<td>2</td>
<td>1005.1 (107.6)</td>
<td>1246.1 (120)</td>
<td>918.9 (99)</td>
<td>936.8 (120.6)</td>
<td>1240.9 (124.5)</td>
<td>1363.6 (99.8)</td>
</tr>
<tr>
<td>3</td>
<td>1299 (109.2)</td>
<td>1831.8 (107.8)</td>
<td>2283.8 (131.6)</td>
<td>2862.1 (107.9)</td>
<td>1830.4 (108.6)</td>
<td>2338 (95.7)</td>
</tr>
<tr>
<td>4</td>
<td>3668.7 (116.2)</td>
<td>2517.3 (102.7)</td>
<td>2502.3 (105.5)</td>
<td>3272.9 (108.7)</td>
<td>3033.8 (110)</td>
<td>2798.7 (116.5)</td>
</tr>
<tr>
<td>5</td>
<td>4290.5 (146.6)</td>
<td>3740 (121.2)</td>
<td>3360.1 (130.5)</td>
<td>3677.1 (105.7)</td>
<td>3775 (101.3)</td>
<td>3197.5 (115.5)</td>
</tr>
</tbody>
</table>

Table 5. Mean and standard deviation (in parenthesis) of each formant for each phoneme.

In Figures 8[13] we represent in time and frequency a phoneme from the database and a pattern-generated phoneme. As one can see, their frequency response is quite similar. Finally, we add that these artificial phonemes are successfully classified by the neural network.
6.6. PHONEME GENERATOR

Fig. 9. Figures of the phoneme /e/.

(a) Spectrum of the generated phoneme.  
(b) Spectrum of a database sample

(c) Time domain plot of the generated phoneme.  
(d) Time domain plot of a database sample

Fig. 10. Figures of the phoneme /i/.

(a) Spectrum of the generated phoneme.  
(b) Spectrum of a database sample

(c) Time domain plot of the generated phoneme.  
(d) Time domain plot of a database sample
6. SPEECH CLASSIFICATION

Fig. 11. Figures of the phoneme /o/.

Fig. 12. Figures of the phoneme /m/.
6.7. MATLAB functions

In this section we present the main MATLAB functions that we have coded to parse speech into phonemes, classify them and generate new speech renderings. For every function we present the inputs (in lower-case letters), outputs (in capital letters) and a short explanation of what the function does (Except for the Peter Assmann function). To see the whole code, see Appendix C.

6.7.1. Segmentation and classification functions. In this subsection we list the principal MATLAB functions for the segmentation and classification.

- function S=initialize(y)
  - y = the initial signal.
  - S = the signal after changing the sample rate.
  - This function changes the sample rate of the signal and erases the noisy parts of the signal.

- function [E,R,I] = segmentation(s,w)
  - s = signal.
  - w = type of wavelets used.
  - I = points of segmentation.
  - E = envelope of each subband after the wavelet transform.
  - R = derivative of the power of each subband.
  - Returns the points of possible segmentation and the function of power and its derivative for each subband. We apply the method explained in 6.2.1.
• function \([L,T]=\text{readdatabase}(p)\)
  - \(p\) = path of the database.
  - \(L\) = list of database phoneme signals.
  - \(T\) = array with the number of samples of each phoneme.
  - Read all the database phonemes of a given path.

• function \([N,T]=\text{neuralnetwork}(i,o,t)\)
  - \(i\) = training inputs.
  - \(o\) = training outputs.
  - \(t\) = training test.
  - \(N\) = neural network after training.
  - \(T\) = training test outputs.
  - This function trains a neural network to classify the different phonemes. It also returns the training test output. To create the neural network we use the \texttt{Matlab} function \texttt{newff}.

• function \([P]=\text{phonemeparameters}(s,i)\)
  - \(s\) = signal.
  - \(i\) = points of segmentation.
  - \(P\) = parameters of each phoneme.
  - It calculates the first 5 formants of each phoneme as explained in 6.3.1. The most important \texttt{Matlab} functions are \texttt{lpc} and \texttt{roots}.

• function \([P]=\text{simulate\_nn}(n,p)\)
  - \(n\) = neural network used.
  - \(p\) = parameters of the phoneme.
  - \(P\) = phoneme after the classification.
  - It takes the trained neural network and the parameters from a phoneme and outputs its classification. The principal function is \texttt{sim}.

6.7.2. Generation and synthesis functions. In this subsection we present the principal functions for the generation of sounds.

• function \([A,P]=\text{get\_vspect}(f0,ft,bt,nft,r)\)
  - \(f0\) = fundamental frequency.
  - \(ft\) = formants of the phoneme.
  - \(bt\) = bandwidth of each phoneme.
  - \(nft\) = number of formants.
  - \(r\) = sample rate of the spectrum signal.
  - \(A\) = amplitude of the generated phoneme spectrum.
  - \(P\) = phase of the generated phoneme spectrum.
  - Returns the spectrum of the generated phoneme from its formants. This function is due to Peter Assmann.

• function \(Y=\text{synthesise}(d,r,ft,f0)\)
  - \(d\) = duration of the phoneme synthesis.
  - \(r\) = frequency rate of the synthesis.
  - \(ft\) = formants of the phoneme.
  - \(f0\) = fundamental frequency of the signal.
  - \(Y\) = generated sound.
– Returns the generated phoneme from its formants. To do it, first applies the `get_vspect` function presented above and then passes its spectrum to the time domain.

- function `[M,S]=phonemetype(p,t)`
  - `p` = parameters of the database phonemes.
  - `t` = list of phonemes of each class.
  - `M` = mean of each formant pattern phoneme.
  - `S` = standard deviation of each formant pattern phoneme.
  - It calculates the mean and standard deviation of an hypothetical ideal phoneme. It follows the algorithm described in [6.6](#).
Conclusions and future work

In this project we have presented an introduction to some signal processing tools, with emphasis in wavelets and pattern theory. We have also illustrated their potential by experimenting with two proof-of-concept applications in image and voice processing, respectively.

The underlying idea of pattern theory is to produce classes of stochastic models that can describe the patterns we see in nature, in the sense that random samples from these models have the same look and feel as the samples from the world itself. The detection of patterns in a real signal can be achieved by finding the pattern sample that is closest to it. This can be done by means of Bayes’ theory. There are two key features of this approach that are worth mentioning. On one hand, recognition can be successful even for quite noisy and ambiguous samples. On the other, the pattern model allows the production of endless variations by playing with its stochastic parameters or with its structural elements.

As we have described in the text, wavelet analysis is a relatively new tool that generalizes Fourier analysis. The Fourier transform decomposes a signal into a sinusoidal basis of different frequencies. As a sine wave has an infinite time duration and specific frequency, it is perfectly localized in frequency domain but not in time domain. To counter this problem, the wavelet transform decomposes a signal in a base formed by shifted and scaled versions of functions limited in both time and frequency.

We have used this local analysis of time signals to extract information from speech signals and parse a recorded voice into phonemes. The idea behind this method is that the power variations of phonemes take place in a narrow band only, so that it is much easier to detect them by analyzing the components of the DWT than by taking into account the power of the whole signal. With this approach we have been able to search for characteristic phoneme patterns, in the sense that they should suffice to distinguish one phoneme from the others. The recognition performance we have reached has more than 70% match to each phoneme used.

As for possible variations, we have shown how to create new voices by tinkering with the parameters of the database.

In the case of images, our goal was to produce a system that on one hand can generate plane mosaics of any of the 17 possible symmetry patterns that these structures can have and, on the other, that can recognize these patterns and hence be able to classify any mosaic by processing an image of it. For the classification part, the core idea was to look at the correlation of the mosaic image with some suitably chosen transformations of it.
The implementations of the algorithms described in the last two chapters have been coded in MATLAB. The listings can be found in the Appendices B and C.

**Future Work**

There are a number of different possibilities to continue the work undertaken in this project, including some that are inspired by the applications of wavelet analysis and pattern theory that have arisen during its development.

We would like to improve our speech synthesiser/recogniser explained in the Chapter 6. In particular, it would be interesting to try it with the whole Catalan phoneme alphabet, and with more voice samples, in order to improve our neural network matches. We would also like to improve our phoneme model in order to make possible not only to experiment with variations of a voice due to factors such as age or current conditions in the vocal tract, but also to create other voices that are recognizably different and yet native. And instead of voice, a similar research can be undertaken for the case of musical instruments.

In the case of plane mosaics, we can improve our program by implementing a more complex image generator. For example, we could describe the parameters that characterize a decoration style and hence create new samples of that style.

In the pattern theory field, a promising and ambitious topic is its application in the so-called image grammars. These are data structures (parse graphs) to be implemented in systems aimed at ascertaining the most probable interpretation of any given image. A parse graph hierarchically structures the parts of an image (like objects, people or background) and their relations. In some way, it is an ideal arena for the synthesis/analysis interplay at work in pattern theory.

Among the promising applications of pattern theory to image processing, we want to mention the case of biomedical images. In this vast area there is an increasing need for fast and reliable processing that can help clinical and radiology researchers and practitioners. One goal here is to model the patterns of disease lesions as they appear in the images, as for example lesions of multiple sclerosis in nuclear magnetic resonance. Another goal is to combine different images of the same source in order to obtain the best possible join information.
Appendix A
Neural Networks

In this appendix we outline an introduction to artificial neural networks (ANN). Neural networks try to mimic how our own brains work, admittedly in a rather limited way, and they have had a big recent resurgence because of advances in computer hardware. See [46].

An ANN is a processing system that consists of elements with local computing capacity (neurons), organized into layered structures that are interconnected and work cooperatively. Neural networks are used because they are capable of learning.

Given a specific task to solve, and a class of functions \( F \), learning means using a set of observations to find \( f^* \in F \) which solves the task in some optimal sense. The steps to solve a problem with a neural network are:

- Learning phase
- Testing phase
- Simulation

The learning phase has the following parts:

- A training set formed by examples of inputs and their expected outputs.
- A cost function that estimates the separation between the model and the concept that we try to model with the training set.
- A optimization algorithm, which normally is the backpropagation algorithm.
- A stopping criterion that allows to control the training time.

The testing phase uses some examples, the test set (normally different from the training set), to check the quality of the learning phase and whether it is necessary to redo it. Finally, once the system has been trained and tested properly, we can use the system to calculate the output of the desired inputs.

An example of an ANN is a Multi Layer Perceptron (MLP). The MLP is an ANN consisting of multiple layers. This allows to solve problems that are not linearly separable. This type of ANN is used in Chapter 6 and implemented in MATLAB.

The layers of a MLP can be classified into three types:
• Input layer: Consisting of neurons that accept input features and pass them to the network. In these neurons there is no processing.
• Hidden layers: Consisting of those neurons whose inputs come from previous layers and whose outputs are passed to the following layers.
• Output layer: Neurons whose output values are the outputs of the entire network.

The main advantage of a MLP is its ease of use, and that it can approximate any input-output examples. The main drawback is its slow training and the high number of training samples required.

The tasks ANNs are applied to belong principally to the following areas:

• Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.
• Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
• Data processing, including filtering, clustering, blind source separation and compression.
• Robotics, including the managing of manipulators, and computer numerical control.
Appendix B

Mosaic’s Code

In this appendix we list the code of the mosaic’s algorithm. In the larger functions we only include the most important part of the code.

- function [I]=imgenerador

```matlab
sI=1000;
I=zeros(sI,sI,3,'uint8');
a=random('unid',256);
if a==256
    a=0;
end
b=random('unid',256);
if b==256
    b=0;
end
c=random('unid',256);
if c==256
    c=0;
end
I(:,:,1)=a;
I(:,:,2)=b;
I(:,:,3)=c;
d=random('unid',9);
if d==9
    d=0;
end
k=1;
while k<=d
    bool=0;
    di=random('unid',sI-40)+20;
dj=random('unid',sI-40)+20;
mval=min([sI-di sI-dj di dj]);
    for i=20:sI-20
        if ((di-i)*(di-i)+(dj-i)*(dj-i)<400) || ((di-i)*(di-i)+(dj-(sI+1-i))*(dj-(sI+1-i))<400)
            bool=1;
        else
            mval=min([sqrt((di-i)*(di-i)+(dj-i)*(dj-i)) mval]);
        end
    end
    k=k+1;
end
end
```

81
\begin{verbatim}
end
if bool==0
    mval=floor(mval);
    r=random('unid',mval-10)+10;
    if mod(r,2)==0
        r=r+1;
    end
    r=(r-1)/2;
    da=random('unid',256);
    if da==256
        da=0;
    end
    db=random('unid',256);
    if db==256
        db=0;
    end
    dc=random('unid',256);
    if dc==256
        dc=0;
    end
    for i=-r:r
        for j=-r:r
            if(i*i+j*j<=r*r)
                I(di+i,dj+j,1)=da;
                I(di+i,dj+j,2)=db;
                I(di+i,dj+j,3)=dc;
            end
        end
    end
    k=k+1;
end
end
figure(1)
imshow(I);
\end{verbatim}

In the next function we have excluded some type of cells whose code is similar to the code of other cells.

- \textbf{function M=cellcreation(i)}

\begin{verbatim}
    X=i;
    tip=random('unid',17);
    if tip==1
        M=X;
    elseif tip==2
        X=imresize(,[size(X,1) 2*size(X,2)]);
        M=[X; flipdim(flipdim(X,2),1)];
    elseif tip==3
        X=imresize(X,[size(X,1) 2*size(X,2)]);
        M=[X; flipdim(X,1)];
    (...)
    elseif strcmp tip==10
        M=[X flipdim(transmat(X,2))];
        M=[M; flipdim(flipdim(M,2),1)];
    elseif strcmp tip==11
        for i=1:size(X,1)
\end{verbatim}
for j={i+1:size(X,2)}
    X(i,j,:) = X(j,i,:);
end

M = [X flipdim(transmat(X),2)];
M = [M; flipdim(flipdim(M,2),1)];

elseif tip == 12
for i=1:size(X,1)-1
    for j=1:size(X,2)-i
        X(size(X,1)-j, size(X,2)-i,:) = X(i,j,:);
    end
end
M = [X flipdim(transmat(X),2)];
M = [M; flipdim(flipdim(M,2),1)];

elseif strcmp tip == 17
    A = [cosd(-30) -sind(-30) 0; 0 1 0; 0 0 1];
    trans = maketform('affine', A);
    J = imtransform(X, trans);
    for i=1:size(J,1)
        for j=1:size(J,2)
            if (size(J,2)*i-(size(J,1)-2*size(X,1))*j>size(X,1)*size(J,2))
                J(i,j,:) = 0;
            end
        end
    end
    L = J;
    for i=1:size(J,1)
        for j=1:size(J,2)
            if (346*i-599*j<0)
                L(i,j,:) = 0;
            end
        end
    end
    M = [zeros(size(J)) L zeros(size(J))];
    B = [cosd(-120) sind(-120); -sind(-120) cosd(-120)];
    v = [size(X,1) 0]*B;
    B = [B [0;0]; [v 1]];
    trans2 = maketform('affine', B);
    LT = imtransform(L, trans2);
    dif = floor((size(LT,2)-size(L,2))/2);
    M(:,1:size(L,2),:) = M(:,1:size(L,2),:) + LT(1:size(L,1),dif+(1:size(L,2)),:);
    LT = flipdim(LT,2);
    M(:,1:size(L,2),:) = M(:,1:size(L,2),:) + LT;
    B = [-cosd(60) -sind(60); -cosd(30) sind(60)];
    v = [-size(M,1) 0]*B + [size(M,1) 0];
    B = [B [0;0]; [v 1]];
    B2 = [cosd(60) sind(60); -sind(60) cosd(60)];
    v = [floor(size(M,1)/2) floor(size(M,2)/2)]*B2;
    B2 = [B2 [0;0]; [v 1]];
    B = B*B2;
    trans2 = maketform('affine', B);
    MT = imtransform(M, trans2);
    dif = floor((size(MT,1)-size(M,1))/2);
\[ M(:,1:\text{size}(M,2),:)=M(:,1:\text{size}(M,2),:)+M\text{(dif+(1:\text{size}(M,1)),:,:)}; \]
\[ M=\text{flipdim(}\text{flipdim}(M,2),1); \]
\[ M=M+MT; \]
\[ A=[1 \ 0 \ 0; \cosd(120) \ \text{sind}(120) \ 0; 0 \ 0 \ 1]; \]
\[ A=A^{-1}; \]
\[ \text{trans=maketform('affine',A);} \]
\[ JM=\text{imtransform}(M, \text{trans}); \]
\[ \text{dif=floor((size}(JM,2)-\text{size}(JM,1))/2); \]
\[ M=JM(:,\text{dif+(1:\text{size}(JM,1)),:}); \]

\[ \text{figure}(3) \]
\[ \text{imshow}(M); \]

\* function \[ [M]=\text{tiling}(c,t) \]

\[ X=c; \]
\[ \text{if} \ \text{tip}<=2 \]
\[ \quad a=\text{random('unif',1,3)}; \]
\[ \quad b=\text{random('unif',1,3)}; \]
\[ \quad \text{angle}=\text{random('unif',30,150)}; \]
\[ \text{elseif} \ \text{tip}<=4 \]
\[ \quad a=\text{random('unif',1,3)}; \]
\[ \quad b=\text{random('unif',1,3)}; \]
\[ \quad \text{angle}=90; \]
\[ \text{elseif} \ \text{tip}==5 \]
\[ \quad a=\text{random('unif',1,3)}; \]
\[ \quad b=a; \]
\[ \quad \text{angle}=\text{random('unif',30,150)}; \]
\[ \text{elseif} \ \text{t}<=8 \]
\[ \quad a=\text{random('unif',1,3)}; \]
\[ \quad b=\text{random('unif',1,3)}; \]
\[ \quad \text{angle}=90; \]
\[ \text{elseif} \ \text{t}==9 \]
\[ \quad a=\text{random('unif',1,3)}; \]
\[ \quad b=a; \]
\[ \quad \text{angle}=\text{random('unif',30,150)}; \]
\[ \text{elseif} \ \text{t}<=12 \]
\[ \quad a=\text{random('unif',1,3)}; \]
\[ \quad b=a; \]
\[ \quad \text{angle}=90; \]
\[ \text{else} \]
\[ \quad a=\text{random('unif',1,3)}; \]
\[ \quad b=a; \]
\[ \quad \text{angle}=60; \]
\[ \text{end} \]
\[ n=\text{size}(X,1); \]
\[ r=200; \]
\[ A=[r/n \ 0 \ 0; \ r/n \ 0; 0 \ 0 \ 1]; \]
\[ \text{figure}(1) \]
\[ \text{image}(X); \]
\[ \text{trans=maketform('affine',A);} \]
\[ [X]=\text{imtransform}(X, \text{trans}); \]
\[ XX=[]; \]
\[ \text{num}=5; \]
\[ \text{for} \ i=1:\text{num} \]
aux=[];
for j=1:num
    aux=[aux X];
end
XX=[XX;aux];
end
angle2=180-angle;
A=[b 0 0; a*cosd(angle2) a*sind(angle2) 0; 0 0 1];
trans=maketform('affine',A);
[J,xdata,ydata]=imtransform(XX, trans,'XYScale',1);
disp(xdata)
disp(size(J));
figure(4)
imshow(J)
if (angle==90)
    J=J;
elseif (angle<90)
    for i=1:size(J,2)
        if(J(1,i,1)˜=0)||(J(1,i,2)˜=0)||(J(1,i,3)˜=0)
            xv=i;
            break;
        end
    end
    J=J(:,xv:size(J,2)-xv,:);
else
    for i=0:size(J,2)-1
        if(J(1,size(J,2)-i,1)˜=0)||(J(1,size(J,2)-i,2)˜=0)||(J(1,size(J,2)-i,3)˜=0)
            xv=i;
            break;
        end
    end
    J=J(:,xv:size(J,2)-xv,:);
end
figure(2)
image(J)

function [X,A]=mosaics(i)

nbcol = 255;
n=1;
XG=rgb2gray(i);
X=im2double(XG);
for i=1:n
    [cA1,cH1,cV1,cD1] = dwt2(X,'db1');
    X = wcodemat(cA1,nbcol);
end
XGG=X-mean(mean(X));
XD=[XGG zeros(size(XGG,1),size(XGG,2));zeros(size(XGG,1),2*size(XGG,2))];
a=ifft2(fft2(XD).*conj(fft2(XD)));
a=a-min(min(real(a)));
a2=abs(a/max(max(a)));
acorr=fftshift(a2);
XGG=zeros(size(X,G,1),size(X,G,2))+0.5;
XD=[XGG zeros(size(X,G,1),size(X,G,2));zeros(size(X,G,1),2*size(X,G,2))];
a=ifft2(fft2(XD).*conj(fft2(XD))); 
a=a-min(min(real(a))); 
a2=abs(a/max(max(a))); 
b=fftshift(a2); 
acorr=acorr./b; 
acorr=acorr(cell(size(b,1)/4):1:floor(3*size(b,1)/4),cell(size(b,2)/4):1:floor(3*size(b,2)/4)); 
A=acorr/max(max(acorr));

• function [P,V,R]=maxims(a,x)

rmin=20; 
n=5; 
resol=0.6; 
posf=2; 
l1=size(A,1); 
l2=size(A,2); 
M=zeros(floor(l1/n),floor(l2/n)); 
ind=M; 
for j=n+1:1:floor(l2/n)-1 
    for i=n+1:1:floor(l1/n)-1 
        M(i+1,j+1)=max(max(A(i*n+1:1:(i+1)*n,j*n+1:1:(j+1)*n))); 
        f=find(M(i+1,j+1)==A(i*n+1:1:(i+1)*n,j*n+1:1:(j+1)*n)); 
        s=posicio(f(1),n); 
        ind(i+1,j+1)=s(1)+i*n+(j*n+s(2)-1)*l1; 
    end 
end 
l1=floor(l1/n); 
l2=floor(l2/n); 
MM=[]; 
indd=[]; 
for j=0:1:l2-3 
    for i=0:1:l1-3 
        m=max(max(M(i+1:1:i+3,j+1:1:j+3))); 
        if(M(i+2,j+2)==m) 
            MM=[MM m]; 
            indd=[indd ind(i+2,j+2)]; 
        end 
    end 
end 
indout=[]; 
for i=1:1:size(MM,2) 
    if(MM(i)>resol) 
        indout=[indout [ind(i+1,i+1);MM(i)]]; 
    end 
end 
indout=transpose(sortrows(transpose(indout),2)); 
indout=indout(:,size(indout,2):-1:1); 
TT=[indout(:,1):10000000000]; 
for i=1:1:size(indout,2) 
    daux=1000000000000000; 
    posi=posicio(indout(1,i),size(A,1)); 
    for j=1:1:1 
        posj=posicio(indout(1,j),size(A,1)); 
        if distmat(posi,posj)<daux 
            daux=distmat(posi,posj); 
        end 
    end 
end
end
end
TT=TT(end,:); daux;
end
TT=transpose(sortrows(transpose(TT),3));
TT=TT(:,size(TT,2):-1:1);
nmax=min([max(nmin,4*ceil(size(TT,2)/5) size(TT,2) 80]));
indout=TT(1:2,1:nmax);
mid=[floor(size(A,1)/2) floor(size(A,2)/2)];
dis=[];
for i=1:1:size(indout,2)
    pos=posicio(indout(1,i),size(A,1));
dis=[dis distmat(mid,pos)];
end
punts=[];
m=find(min(dis)==dis);
for i=1:1:size(indout,2)
punts=[punts;indout(1,i) distmat(posicio(indout(1,i),size(A,1)),
posicio(indout(1,m),size(A,1))) posicio(indout(1,i),size(A,1))
    -posicio(indout(1,m),size(A,1)) indout(2,i)];
end
P=sortrows(punts,2);
v1=P(2,:);
for i=3:1:size(P,1)
    if (abs(cosd(vectangle(v1(3:1:4),punts(i,3:1:4))))<0.95)&
        ((vectangle(v1(3:1:4),punts(i,3:1:4))<117)||(vectangle(v1(3:1:4),punts(i,3:1:4))>123))
        v2=punts(i,:);
        break;
    end
end
V=[v1;v2];
angle=acosd(abs((v1(3)*v2(3)+v1(4)*v2(4))/sqrt(v1(2)*v2(2))));
A=[cosd(60) sind(60);-sind(60) cosd(60)];
vv=A*transpose(v1(3:4));
if (v1(2)/v2(2)>0.95)
    mvect=1;
else
    mvect=0;
end
if (mvect==1)&&(angle>57)&&(angle<63)
    R=5;
elseif (mvect==1)&&(angle>87)
    R=4;
elseif (mvect==1)
    R=3;
elseif (angle>87)
    R=2;
else
    R=1;
end
if (tip==5)&&(acosd(abs((vv(1)*v2(3)+vv(4)*v2(4))/sqrt(v1(2)*v2(2)))))<2)
    V=[v2;v1];
end

function pos=posicio(n,dim)
if(n==floor(n/dim)*dim)
    pos=[dim floor(n/dim)];
else
    pos=[n-floor(n/dim)*dim floor(n/dim)+1];
end

function dis=distmat(a,b)
    dis=(a(1)-b(1))*(a(1)-b(1))+(a(2)-b(2))*(a(2)-b(2));
end

function ang=vectangle(a,b)
    ang=acosd(dot(a,b)/sqrt(dot(a,a)*dot(b,b)));
end

function [A,J,M]=squarecell(i,v)
    v1=v(1,:);
    v2=v(2,:);
    num=49;
    A1=[v2(2) v1(2);v2(1) v1(1)];
    A=A1^(-1)*(num+1);
    A=[A [0;0];0 0 1];
    A=transpose(A);
    trans=maketform('affine',A);
    [J,xdata,ydata]=imtransform(1, trans);
    dim=size(1,1);
    if mod(dim,2)==0
        dim=dim+1;
    end
    mig=posicioplot(punt,dim);
    M=floor(tformfwd(trans,mig(1),mig(2))-[xdata(1) ydata(1)]);
end

function pos=posicioplot(n,dim)
    if(n==floor(n/dim)*dim)
        pos=[floor(n/dim) dim];
    else
        pos=[floor(n/dim)+1 n-floor(n/dim)*dim];
    end
end

In the next function we only present the case of a 6-fold rotation. The other cases are similar.

function [J,V]=rotation(i,t,v)
    num=49;
    [A1,J1,migt]=squarecell(i,v);
    I=J1(migt(2)+(0:1:num),migt(1)+(0:1:num));
    (...) elseif t==6
        [A2,J2,migt2]=squarecell(i,v(2,:),v(2,:)-v(1,:));
        J2=J2(migt2(2)+(0:1:num),migt2(1)+(0:1:num));
        gr=6;
    end
    if mod(size(a31,1),2)==0
        dim=size(a31,1)+1;
    else
        dim=size(a31,1);
    end
mid = [floor(size(I,1)/2) floor(size(I,2)/2)];
t = [1-cosd(gr) sind(gr); -sind(gr) 1-cosd(gr)]*[mid(2);mid(1)];
I1 = I-mean(mean(I));
I2 = 3-mean(mean(I));
Icorr = ifft2(fft2(I2).*conj(fft2(I1)));
a = max(max(fft2(I2).*conj(fft2(I1))));
Icorr = Icorr/a;
V = max(max(Icorr));
mm = find(V == Icorr);
xx = posicio(mm, size(Icorr,1));
R = [cosd(gr) sind(gr) 0; -sind(gr) cosd(gr) 0; t(1) t(2) 1];
T = [1 0 0 1 0; xx(2) xx(1) 1];
A = A1*T*A1^(-1)*R;
B = [1 0 0 1 0; 0 0 1];
v = [-A(3,1)+B(3,1); -A(3,2)+B(3,2)];
M = [A(1,1)-B(1,1) A(2,1)-B(2,1); A(1,2)-B(1,2) A(2,2)-B(2,2)];
x = M^(-1)*v;
trans2 = maketform('affine', A1);
a = floor(tformfwd(trans2, transpose(x)));
xx = floor(a-[xdata(1) ydata(1)]);
x = floor(transpose(xx));
figure(1)
image(Ji(xx(2)+(0:1:num), xx(1)+(0:1:num)));

function a = invposicio(b,s1)
a = (b(2)-1)*s1-b(1);
B. Mosaic's Code

```matlab
mm = find(bmax == b(:,1));
aux = bmax/a;
if aux > V
    V = aux;
    G = mm - 1;
end
elseif t == 2
    I = i;
    I2 = transpose(I);
    I = I - mean(mean(I));
    I2 = I2 - mean(mean(I2));
    a = max(max(ifft2(fft2(I).*conj(fft2(I)))));
    b = ifft2(fft2(I2).*conj(fft2(I)));
    bmax = 0;
    for i = 1:1:num + 1;
        if bmax < b(i,i)
            bmax = b(i,i);
            mm = i;
        end
    end
    aux = bmax/a;
    if aux > V
        V = aux;
        G = mm - 1;
    end
else
    I = i;
    I2 = I(size(I,1):-1:1,:);
    I2 = transpose(I2);
    I2 = I2(size(I,1):-1:1,:);
    I = I - mean(mean(I));
    I2 = I2 - mean(mean(I2));
    a = max(max(ifft2(fft2(I).*conj(fft2(I)))));
    b = ifft2(fft2(I2).*conj(fft2(I)));
    bmax = 0;
    for i = 1:1:num + 1;
        if bmax < b(i,i)
            bmax = b(i,num + 2 - i);
            mm = i;
        end
    end
    aux = bmax/a;
    if aux > V
        V = aux;
        G = mm - 1;
    end
end
end
```
Appendix C
Speech Code

In this appendix we display the code of the speech methods explained in Chapter 6.

- function \( S = \text{initialize}(y) \)

\[
[c1,1l]=\text{wavedec}(y,8,'db8');
c1=c1(1:sum(1l(1:6)));
1l=[1l(1:6);\text{size}(c1,1)];
S = \text{waverec}(c1,1l,'db8');
\]

- function \([E,R,I] = \text{segmentation}(s,w)\)

\[
s=s/\max(s);
\text{numlev}=6;
[c,l]=\text{wavedec}(s,\text{numlev},w);
n=0;
slev=\text{zeros}(\text{numlev},1);
\text{for } i=1:\text{numlev}
\quad n=\max(n,\text{ceil}(l(i+1)/2^{(i-1)}));
\quad slev(i)=\text{ceil}(l(i+1)/2^{(i-1)});
\text{end}
p=\text{zeros}(n,\text{numlev});
r=\text{zeros}(n+3,\text{numlev});
\text{env}=p;
h=[1 2 -2 -1];
\text{for } i=1:\text{numlev}
\quad cp=\text{zeros}(n*2^{(i-1)},1);
\quad cp(1:1+i)=c(1+\text{sum}(l(1:i))):\text{sum}(l(1:i+1)));
\quad res=\text{zeros}(n,1);
\quad \text{for } j=1:2^{(i-1)}
\quad \quad res=\text{res}+cp(j:2^{(i-1)}:n*2^{(i-1)}).*cp(j:2^{(i-1)}:n*2^{(i-1)});
\quad \quad \text{end}
\quad p(:,i)=res;
\quad r(:,i)=\text{abs}(\text{conv}(p(:,i),h))/6;
\text{end}
\text{for } j=1:\text{numlev}
\quad t=\text{floor}(w(j)/2);
\quad \text{for } i=t+1:n
\quad \quad \text{if } i<t+n+1
\text{end}
\]
env(i,j)=max(p(i-t:i+t,j));
else
env(i,j)=env(i-1,j);
end
end
for i=t:-1:1
env(i,j)=max(p(1:i+t));
end
index=zeros(n,numlev);
nindex=zeros(1,6);
for j=1:numlev
pp=0.02;
pmin=0.003;
ind=[];
for i=2:n-1
if (abs(r(i,j)-env(i,j))<pp)&&((abs(r(i+1,j)-env(i+1,j))>pp)||
(abs(r(i-1,j)-env(i-1,j))>pp))&&env(i,j)>pmin
ind=[ind i];
end
end
nindex(j)=size(ind,2);
index(1:size(ind,2),j)=ind';
end
in=max(nindex);
aa=[];
for i=1:numlev
aa=[aa index(1:in,i)'/slev(i)];
end
aa=sort(aa);
aa=aa(find(and(aa>0,aa<1)));
t=0.030;
fs=11025;
delta=t*fs/size(s,1);
aam=[];
summ=aa(1);
aux=1;
j=1;
for i=2:size(aa,2)
if i==size(aa,2)
if aa(i)-aa(aux)>3*delta
mea=(aa(i)+aa(aux))/2;
tind=find(aa>mea);
tind=tind(1);
if (tind==i)&&(aa(i)-aa(i-1)>delta)
aam=[aam sum(aa(aux:tind-1))/(tind-aux) aa(i)];
elseif aa(i)-aa(i-1)>delta
aam=[aam sum(aa(aux:tind-1))/(tind-aux) sum(aa(tind:i-1))/(i-tind) aa(i)];
else
aam=[aam sum(aa(aux:tind-1))/(tind-aux) sum(aa(tind:i))/(i+1-tind)];
end
j=1;
summ=aa(i);
aux=i;
elseif aa(i)-aa(i-1)>delta
C. SPEECH CODE

\[ \text{summ} = \frac{\text{summ}}{j}; \]
\[ \text{aam} = [\text{aam}, \text{summ}, \text{aa}(i)]; \]
\[ \text{else} \]
\[ \text{summ} = \frac{\text{summ} + \text{aa}(i)}{(j+1)}; \]
\[ \text{aam} = [\text{aam}, \text{summ}]; \]
\[ \text{end} \]
\[ \text{else} \]
\[ \text{if} \ \text{aa}(i) - \text{aa}(\text{aux}) > 3*\text{delta} \]
\[ \text{if} \ \text{(aa}(i) - \text{aa}(i-1) > \text{delta}) \&\& \text{(aa}(i) - \text{aa}(\text{aux}) > 2.4*\text{delta}) \]
\[ i = i - 1; \]
\[ \text{mea} = \frac{\text{aa}(i) + \text{aa}(\text{aux})}{2}; \]
\[ \text{tind} = \text{find}(\text{aa} > \text{mea}); \]
\[ \text{aam} = [\text{aam}, \text{sum}(\text{aa}(\text{aux}:\text{tind}-1))/(\text{tind}-\text{aux}) \text{ sum}(\text{aa}(\text{tind}:i))/(i+1-\text{tind})]; \]
\[ i = i + 1; \]
\[ j = 1; \]
\[ \text{summ} = \text{aa}(i); \]
\[ \text{aux} = i; \]
\[ \text{elseif} (\text{aa}(i) - \text{aa}(i-1) > \text{delta}) \]
\[ \text{summ} = \frac{\text{summ}}{j}; \]
\[ \text{aam} = [\text{aam}, \text{summ}]; \]
\[ j = 1; \]
\[ \text{summ} = \text{aa}(i); \]
\[ \text{aux} = i; \]
\[ \text{else} \]
\[ \text{mea} = \frac{\text{aa}(i) + \text{aa}(\text{aux})}{2}; \]
\[ \text{tind} = \text{find}(\text{aa} > \text{mea}); \]
\[ \text{tind} = 1; \]
\[ \text{aam} = [\text{aam}, \text{sum}(\text{aa}(\text{aux}:\text{tind}-1))/(\text{tind}-\text{aux}) \text{ sum}(\text{aa}(\text{tind}:i))/(i+1-\text{tind})]; \]
\[ j = i + 1 - \text{tind}; \]
\[ \text{summ} = \text{sum}(\text{aa}(\text{tind}:i)); \]
\[ \text{aux} = \text{tind}; \]
\[ \text{end} \]
\[ \text{elseif} \ \text{aa}(i) - \text{aa}(i-1) > \text{delta} \]
\[ \text{summ} = \frac{\text{summ}}{j}; \]
\[ \text{aam} = [\text{aam}, \text{summ}]; \]
\[ j = 1; \]
\[ \text{summ} = \text{aa}(i); \]
\[ \text{aux} = i; \]
\[ \text{else} \]
\[ \text{summ} = \text{sum} + \text{aa}(i); \]
\[ j = j + 1; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{I} = \text{aam}; \]
\[ \text{E} = \text{env}; \]
\[ \text{R} = \text{r}; \]

\[ \bullet \ [\text{L}, \text{T}] = \text{readdatabase}(\text{p}) \]

\text{phon} = ['o', 'ea', 'e', 'i', 'n', 'm'];
\text{sphon} = [1 2 1 1 1 1];
\text{L} = [];
\text{T} = [];
temp=0.02;
for i=1:size(sphon,2)
    pwd=[p phon(1+sum(sphon(1:i-1)):sum(sphon(1:i)))];
    pp=ls(pwd);
    T=[T size(pp,1)-2];
    for j=3:size(pp,1)
        [listaux,f1,n1]=wavread([pwd '/' pp(j,:)]);
        L=[L listaux];
    end
end

• [N,T]=neuronalnetwork(i,o,t)

ival=10;
jval=6;
N=newff(t,o,[ival jval], {'tansig','tansig'});
N=train(N,i,o);
taux=sim(N,t);
tout=zeros(1,size(taux,2));
for i=1:size(taux,2)
    tout(i)=find(max(taux(:,i))<=taux(:,i));
end
T=tout;

• [P]=phonemeparameters(s,i)

fs=11025;
P=[];
for indexi=1:max(size(i))-1
    signal=s(i(indexi):i(indexi+1))';
    SegLength=floor(.025*fs);
    SP=.4;
    PreEmphFact=-0.98;
    NumOfFormants=5;
    MaxFormantFreq=5000;
    MaxBWFreq=600;
    FreqRes=512;
    sigLength=length(signal);
    NumSeg=fix((sigLength-SegLength+SegLength*SP)/(SegLength*SP));
    signal=real(filter([1 PreEmphFact],1,signal));
    BegPtr=1;
    FCFreq=[];
    FCBW=[];
    hamwin=hamming(SegLength);
    AddP=1;
    for n=1:NumSeg
        segment=signal(BegPtr:(BegPtr+SegLength-1));
        segment=segment.*hamwin;
        segment=[segment;zeros(FreqRes-SegLength),1];
        Candidates=FormantCand(segment);
        FSeg=angle(Candidates)*fs/(2*pi);
        FCBW=abs(log(abs(Candidates))*fs/pi);
        FSeg=FSeg(find(FCBW<MaxBWFreq));
        FSeg=FSeg(find(FSeg<MaxFormantFreq));
while (NumberOfFormants>length(FCseg))
    Candidates=FormantCand(segment,AddP);
    FCseg=angle(Candidates)*fs/(2*pi);
    FCBW=abs(log(abs(Candidates))*fs/pi);
    FCseg=FCseg(find(FCBW<MaxBWFreq));
    FCseg=FCseg(find(FCseg<MaxFormantFreq));
    AddP=AddP+1;
end
if (n>1)
    Formants=distFormant(FCseg,NumberOfFormants,LastFormant);
else
    Formants=distFormant(FCseg,NumberOfFormants);
end
ForMat=Formants.FM;
    chance=1./Formants.dst;
    [a b]=max(chance);
    LastFormant=ForMat(:,b);
    FormantTrack(:,n)=LastFormant;
    AddP=1;
    BegPtr=BegPtr+fix(SegLength*SP);
end
FFF=[];
FFO=[];
for i=1:size(FormantTrack,2)
    FFF=[FFF FormantTrack(:,i)']
end
FFF=sort(FFF);
aux=1;
for i=2:size(FFF,2)
    if i=size(FFF,2)
        FFO=[FFO;mean(FFF(aux:i))]
    elseif (FFF(i)-FFF(aux)<300)&&(FFF(i+1)-FFF(i)<FFF(i)-FFF(i-1))&&(FFF(i+1)-FFF(aux)>300)
        FFO=[FFO;mean(FFF(aux:i-1))]
        aux=i;
    elseif FFF(i)-FFF(aux)>300
        FFO=[FFO;mean(FFF(aux:1-i))]
        aux=i;
    end
end
if size(FFO,1)<5
    a=5-size(FFO,1);
    a=0*(1:a);
    FFO=[FFO;a']
end
P=[P FFO(1:5)];
Amp=Amp(posPhaseIndex);
[Phase sIndex]=sort(Phase);
Amp=Amp(sIndex);
Candidates=Amp.*exp(i*Phase);

function dist=distFormant(Poles,numberOfFormants,LastChosenFormant)
FormantMat=(nchoosek(Poles,numberOfFormants))';
numberOfCands=size(FormantMat,2);
dist.MM=FormantMat;
dist.dst=dist;

• function [P]=simulate(nn(n,p))
phon=['o' 'ea' 'e' 'i' 'n' 'm'];
sphon=[1 2 1 1 1 1];
Vaux=nn(n,p);
Vout=zeros(1,size(Vaux,2));
P=[];
for i=1:size(Vaux,2)
    Vout(i)=find(max(Vaux(:,i))<=Vaux(:,i));
    P=[P ' `' phon(1+sum(sphon(1:Vout(i)-1):sum(sphon(1:Vout(i)))) `'`);
end

• [A,P]=get_vspect(f0,ft,bt,nft,r)

f=0:f0:r/2;
tf=ones(size(f));
T=1/r;
z=exp(i*2*pi*f*T);
fgp=0; bgp=1000;
cp=exp(-2*pi*bgp*T);
bp=2*exp(-pi*bgp*T).*cos(2*pi*fgp*T);
ap=1-bp-cp;
tf=tf.*(ap./(1-bp.*z.ˆ(-1)-cp.*z.ˆ(-2))).*(1-z.ˆ(-1));
fgz=1500; bgz=6000;
cz=exp(-2*pi*bgz*T);
bz=2*exp(-pi*bgz*T).*cos(2*pi*fgz*T);
az=1-bz-cz;
cz=-cz/az;
bz=-bz/az;
az=1/az;
tf=tf.*(az./(1-bz.*z.´(-1)-cz.*z.´(-2)));
c=exp(-2*pi*bt*T);
b=2*exp(-pi*bt*T).*cos(2*pi*ft*T);
a=1-b-c;
for ift=1:nft
    tf=tf.*(a(ift)./(1-b(ift).*z.´(-1)-c(ift).*z.´(-2)));
end
amp=abs(tf)';
phase=angle(tf)';

• function Y=synthesise(d,r,ft,f0);
npts=floor(d*(r/1000));
rmspv=d;
nsampf=npts;
T=1/r;
o=ones(nsampf,1);
bt=[90 110 140 180 220];
y=zeros(npts,1);
ps=zeros(nsampf,floor((r/2)/min(f0))+1);
psi=zeros(1,floor((r/2)/min(f0))+1);
if ft(4)>4000
    nft=3;
    ft=ft(1:3);
    bt=bt(1:3);
elseif ft(5)>4800
    nft=4;
    ft=ft(1:4);
    bt=bt(1:4);
end
i=find(ft(1:nft)<30);
nft=nft-length(i); ft(i)=[]; bt(i)=[];
[amp,ph]=get_vspect(f0,ft,bt,nft,r);

• function [M,S]=phonemetype(p,t)

M=[];
S=[];
for ipar=1:max(size(t))
    m=[]; s=[];
    par=p(:,1+sum(t(1:ipar-1)):1:sum(t(1:ipar)));
    for i=1:size(par,1)
        aux=par(i,:);
        aux=sort(aux);
        temp=1;
        tm=[];
        for j=2:max(size(aux))
            if aux(j)-aux(j-1)>150
                tm=[tm j-temp];
                temp=j;
            elseif aux(j)-aux(temp)>400
                tm=[tm j-temp];
                temp=j;
            end
        end
        tm=[tm j-temp];
        ij=find(tm==max(tm));
        ij=ij(1);
        m=[m mean(aux((1+sum(tm(1:ij-1))):1:sum(tm(1:ij))))];
        s=[s std(aux((1+sum(tm(1:ij-1))):1:sum(tm(1:ij))))];
    end
M=[M;m]
S=[S;s];
References

REFERENCES

41. H.V. Pohl, Introduction to orbifolds, ETH Zürich, 2010.