POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Politecnico di Torino

von KARMAN INSTITUTE FOR FLUID DYNAMICS

Master's Degree Thesis

Experimental study on drone acoustics using Drone-Tracking Beamforming method

Candidate
Lorenzo SCHILLACI

PoliTo's supervisors
R. ARINA
F. AVALLONE

VKI's supervisors
C. SCHRAM
R. ZAMPONI

December 2023

Abstract

Unmanned aerial vehicles (UAVs), usually called "drones", are nowadays becoming very popular as recreational devices, but their capabilities make them potentially exploitable to carry many tasks out more easily or quickly than in the past. However, their employment involves not negligible noise emissions, potentially bothering humans and animals in the operational area. This typical noise must be studied and reduced.

This thesis aims at validating a method, denoted as Drone-Tracking Beamforming, based on the Conventional Beamforming algorithm. The method analyzes data about the field produced by an acoustic source and recorded by an array of phased microphones. It allows to produce a sequence of positions, where the source is localized, by dividing the original signal into multiple chunks. When applied to drone recordings, this algorithm enables the automatic tracking of the aircraft as much as it permits to filter out acoustic reflections affecting the original data.

To get appropriate data inputs for the validation, an array of 64 electret microphones was used to record acoustic signals from different kind of sources, in the same environment (ID2MOVE test hall, Belgium). Part of the work presented is about electrets' calibration, a necessary procedure to produce high quality data using this kind of sensors.

Processed data are then used to show the typical output of a beamforming method: colored source-maps indicating different values of SPL. In these maps, the correct position of a sound source (detected by a motion-capture system mounted in ID2MOVE test hall) is highlighted and compared to its estimated position, to prove the tracking capabilities of the method. A second type of output is produced in order to show the characterization of the acoustic field generated by the subject of the tracking: the SPL values are integrated in limited areas around the source and the results are plotted as spectra.

Both the type of outputs could be used to study acoustic emissions of drones, with the aim of controlling and reducing them. This, in turn, could promote a wider adoption and acceptance of these vehicles, even in densely populated areas.

Acknowledgements

ACKNOWLEDGMENTS

Table of Contents

List of Tables

List of Figures

Acronyms

BPF
Blade Passing Frequency

BVI
Blade Vortex Interaction

BWI
Blade Wake Interaction

CB
Conventional Beamforming

DTB
Drone Tracking Beamforming

FT
Fourier Transform

LEE
Linearized Euler Equations

PSD
Power Spectral Density

QMCS
Qualisys Motion Capture System

QTM
Qualisys Track Manager

ROI

Region Of Integration

SG

Scanning Grid

SPI

Source Power Integration

SPL

Sound Pressure Level

TF

Transfer function

TFM

Transfer function Matrix

UAV

Unmanned Aerial Vehicles

VKI

von Karman Institute

Chapter 1

Introduction

1.1 Outline

This thesis describes the experimental work carried out over the course of four months, spent at the von Karman Institute for fluid dynamics (VKI) in Sint-Genesius-Rode, Belgium, and the following data processing. It is about the validation of a technique for the analysis of the acoustic field emitted by drones in flight, called Drone-Tracking Beamforming (DTB). This method performs an analysis on data recorded by a phased array of electret microphones, previously calibrated with a dedicated procedure.

The document is divided into seven parts.

1. The first chapter serves as an introduction to the main topic of this work, providing an overview of the issues related to drone noise production.

2. The second chapter introduces the fundamental equations used to address acoustic problems and establishes a theoretical foundation that is essential for comprehending the methods employed throughout the document.

3. The third chapter provides a description of the beamforming codes, which were implemented in MATLAB and used to process the data collected during the experiments.

4. The fourth chapter is about the data collection process and provides a detailed description of the ID2MOVE test hall in Nivelles, Belgium, as well as of the tools used during the experimental campaign. It outlines the sequence of actions that constituted the data collection methodology.

5. The fifth chapter contains all the information regarding electrets' calibration procedure. It explains the concepts underlying the entire procedure, from the

collection of calibration data to their processing through a dedicated code. The code generates an output which is applied to the experimental signals, enhancing their quality.

6. The sixth chapter focuses on the processing of the calibrated data and presents the results of the DTB analysis conducted in three different types of experiments. First, a method validation is performed by studying signals produced by monopolar sources. Then, hovering drone signals are analyzed to apply the DTB algorithm to an almost steady source, and finally, the method is applied to a fast-flying drone.

7. The seventh chapter contains conclusions regarding the work carried out and outlines future prospects.

1.2 About drones and their noise

Quadcopter drones are systems belonging to the category of Unmanned Aerial Vehicles (UAV), widely used for recreational purposes, but also capable of completing a wide range of tasks, with the ability to manage various critical issues and the possibility of improving services still carried out by classic means of transport.

Applications of such vehicles that could revolutionize transportation would benefit from their ability to bypass traffic. While everyday delivery services would be advantaged by adopting this approach, it would be especially useful in critical scenarios such as rapid medical transportation, in which swift delivery of item, like organs for transplant procedures, is fundamental. On the other hand, drones are already being used for a wide range of monitoring services, encompassing fire control, wildlife observation and protection, production efficiency improvement in agriculture, as well as security and military purposes.

Considering the growing number of quadcopter drones in use, one of the main concerns about their presence is noise pollution, which can potentially impact human life and activities. In particular, when they operate in urban regions, it is important to consider their psychoacoustic effects, which can aggravate the annoyance perceived by individuals in proximity. A comparison between the estimated psychoacoustic annoyances of different kinds of vehicles is shown in figure 1.1, from which it is evident that these drones are considered among the most annoying devices [1]. Furthermore, a drone can be perceived as a threat by wild animals as much as it can be easily recognized by humans, because of its typical tonal acoustic emissions. Hence, its noise could lead to mission failures, in contexts where the drone must remain hidden.

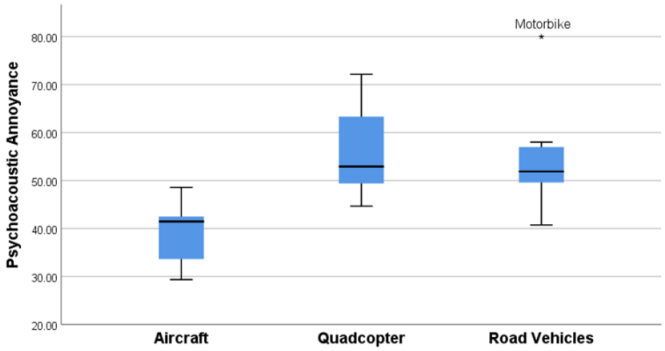

Figure 1.1: Psychoacoustic annoyance for different vehicles

1.3 Objective of the thesis

The objective of this thesis is the validation of a beamforming algorithm capable of tracking a moving drone along its trajectory while characterizing the acoustic field that surrounds it during the maneuvering phases. Acoustic beamforming methods are particularly suitable to fulfill these types of functions. In the past they were used to successfully localize both fast running trains and flying airplanes, using simple linear arrays [2]. However a beamforming method can also be used to analyze the acoustic field of the region inspected, while filtering out environmental reflection contribution, so that the characteristic emissions of a localized moving source can be studied.

Maneuvering phases are responsible for the greatest drones' noise emissions and must be deeply investigated. Direction changes require variations in a drone's attitude and in propellers' rotation speed, which in turn determine variations of the Blade Passing Frequency (BPF), variations of the actions applied on the propellers, the increase in turbulence and its interactions with all surfaces of the device [3] [4].

In order to achieve the objective, an experimental campaign was conducted with two visits at ID2MOVE indoor test zone, a wide facility specifically designed for drones' flight and broad maneuvers. This test-hall is equipped with a motion-capture system, allowing the comparison of DTB method's tracking results with the trajectories detected by ID2MOVE's system. Several recordings were made, with particular attention to the repeatability of the experiments, allowing for their comparisons regarding alterations in specific variables, as the velocity of the subject or its distance from the phased array.

Attention was paid to the electret microphones composing the array. They were sorted and selected depending on the quality of their output and then eventually repaired or replaced. A specific calibration procedure was performed and applied to the whole set of sensor in order to enhance the results of the DTB algorithm.

Chapter 2

Theory background

This chapter contains a description of the theoretical tools that will be used in the thesis. At first the primary relations about fluid dynamics are described, followed by a basic explanation of sound propagation theory and the illustration of the principal tools used for processing of experimental acoustic data.

The theory behind the Conventional Beaforming (CB) method, which is implemented in DTB algorithm, is then explained, followed, at the end, by a description of drones' noise generation mechanisms.

2.1 Fluid-dynamics equations

First of all, an assumption about the fluid is done: it is considered to be a continuum, so that it is possible to define in it a volume particle, with smaller dimensions than any of those characterizing the size of the problem, but still much bigger than every molecule in the fluid [5]. This hypothesis enables the definition of punctual, continuous fluid's properties as particles' mean physical quantities.

2.1.1 Navier-Stokes equations

By studying the fluid as a continuum, it is possible to define its physical properties as the solutions to a set of partial differential equations, describing mass, momentum and energy balances. These equations can be derived both in their integral and differential form, and even applying both the Eulerian and the Lagrangian approach. Using a differential-Eulerian derivation, these equations allow the fluid to be considered as flowing through a steady, permeable control volume. Their quasi-linear form is written below:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = Q_m \tag{2.1}$$

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u} = \nabla\cdot\boldsymbol{\Pi} + \rho\boldsymbol{f} \tag{2.2}$$

$$\rho\frac{\partial E}{\partial t} + \rho\boldsymbol{u}\cdot\nabla E = \nabla\cdot(\boldsymbol{\Pi}\cdot\boldsymbol{u}) + \rho\boldsymbol{f}\cdot\boldsymbol{u} - \nabla\cdot\boldsymbol{q} + Q_w \tag{2.3}$$

Equation 2.1 is the mass balance, also denoted as "continuity equation". In it, ρ is fluid's density, $\boldsymbol{u}$ is velocity vector and Q_m is a term representing a mass injection inside the control volume. Equation 2.2 is the momentum balance, a vectorial relation that shows the stress tensor, $\boldsymbol{\Pi}$, and the vector $\boldsymbol{f}$, which takes into account external forces applied to the control volume. Finally, equation 2.3 is the energy balance, in which appear E, the total energy, sum of internal and kinetic energy, the vector $\boldsymbol{q}$, the heat flux through the volume's walls, and Q_w, heat source inside the control volume.

It is possible to define the tensor $\boldsymbol{\Pi}$ as:

$$\boldsymbol{\Pi} = -p\boldsymbol{I} + \boldsymbol{\tau} \tag{2.4}$$

where p is fluid's pressure, $\boldsymbol{I}$ is an identity matrix and $\boldsymbol{\tau}$ is the viscous stresses' tensor. The latter, in turn, can be written as:

$$\boldsymbol{\tau} = \mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) + \lambda(\nabla\cdot\boldsymbol{u})\boldsymbol{I} \tag{2.5}$$

where μ is the dynamic viscosity and λ is the bulk viscosity.

Another useful relation is the Fourier's equation, which is considered as a model for the heat flux through control volume's walls:

$$\boldsymbol{q} = -k\nabla T \tag{2.6}$$

where k is the thermal conductivity and T is the temperature.

It is, now, possible to substitute relations 2.4, 2.5 and 2.6 in the balance equations (2.1, 2.2 and 2.3) to get the most general formulation of Navier-Stokes equations:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}) = Q_m \tag{2.7}$$

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \nabla\cdot\left[\mu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) + \lambda(\nabla\cdot\boldsymbol{u})\boldsymbol{I}\right] + \rho\boldsymbol{f} \tag{2.8}$$

$$\begin{aligned}\rho\left(\frac{\partial E}{\partial t} + \boldsymbol{u}\cdot\nabla E\right) = -\nabla\cdot(p\boldsymbol{u}) + \nabla\cdot\left\{\left[\mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) + \lambda(\nabla\cdot\boldsymbol{u})\boldsymbol{I}\right]\right\} + \\ +\rho\boldsymbol{f}\cdot\boldsymbol{u} + \nabla\cdot(k\nabla\boldsymbol{T}) + Q_w\end{aligned} \tag{2.9}$$

2.1.2 Euler's equations

Neglecting the effects of viscosity and thermal conductivity, it is possible to find out Euler's equations from the Navier-Stokes:

$$\frac{\partial \rho}{\partial t} + \rho \nabla \cdot \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \rho = Q_m \tag{2.10}$$

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \rho \boldsymbol{f} \tag{2.11}$$

$$\rho \left(\frac{\partial E}{\partial t} + \boldsymbol{u} \cdot \nabla E \right) = -\nabla \cdot (p\boldsymbol{u}) + \rho \boldsymbol{f} \cdot \boldsymbol{u} + Q_w \tag{2.12}$$

Instead of using a balance equation based on the total energy, to characterize the state of a fluid in motion it is often preferred the use of the entropy equation. This relation can be obtained by combining 2.1, 2.2 and 2.3:

$$\rho T \frac{Ds}{Dt} = \rho(\boldsymbol{\tau} \cdot \nabla) \cdot \boldsymbol{u} - \nabla \cdot \boldsymbol{q} + Q_w. \tag{2.13}$$

Therefore, instead of using equation 2.12 it is possible to consider the relation:

$$\rho T \left(\frac{\partial s}{\partial t} + \boldsymbol{u} \cdot \nabla s \right) = Q_w. \tag{2.14}$$

2.2 Sound propagation

Sound propagation is the diffusion of a signal through a medium, being the signal a variation in the physical quantities describing the status of the considered system. As the signal propagates into the medium as a wave, the definition of a wave equation is necessary to fully explore any acoustic problem.

2.2.1 Acoustic wave equation

Euler's equations are the starting point to get a proper wave equation. However, it is particularly difficult to deal with density and entropy variations, so pressure fluctuations are chosen to represent the behaviour of acoustic waves.

The first step to attain the sound wave equation is the linearization of relations 2.10, 2.11 and 2.14. Considering p_0, ρ_0, s_0 and $\boldsymbol{u}_0$ as mean values of pressure, density, entropy and velocity, their fluctuating values can be respectively defined as p', ρ', s' and $\boldsymbol{u}'$. Then, if $\boldsymbol{u}_0 = 0$, the following definitions can be written:

$$\begin{aligned} p(\boldsymbol{x}, t) &= p_0 + p'(\boldsymbol{x}, t), \\ \rho(\boldsymbol{x}, t) &= \rho_0 + \rho'(\boldsymbol{x}, t), \\ s(\boldsymbol{x}, t) &= s_0 + s'(\boldsymbol{x}, t), \\ \boldsymbol{u}(\boldsymbol{x}, t) &= \boldsymbol{u}_0 + \boldsymbol{u}'(\boldsymbol{x}, t) = \boldsymbol{u}'(\boldsymbol{x}, t). \end{aligned} \tag{2.15}$$

Using 2.15, it is possible to obtain the Linearized Euler Equations (LEE), or linear acoustic equations:

$$\frac{\partial \rho'}{\partial t} + \rho_0 \nabla \cdot \boldsymbol{u}' = Q_m \tag{2.16}$$

$$\rho_0 \frac{\partial \boldsymbol{u}'}{\partial t} = -\nabla p' + \rho_0 \boldsymbol{f} \tag{2.17}$$

$$\rho_0 T_0 \frac{\partial s'}{\partial t} = Q_w, \tag{2.18}$$

where Q_m is an unsteady mass injection, $\boldsymbol{f}$ represents a fluctuating force field, often linked to a variable aerodynamic action on a wall, and Q_w is a heat production, bound to entropy's variations.

Once defined the LEEs, the next step to get the acoustic wave equation is the subtraction between the time-derivative of 2.16 and the divergence of 2.17:

$$\frac{\partial^2 \rho'}{\partial t^2} - \nabla^2 p' = \frac{\partial Q_m}{\partial t} - \rho_0 \nabla \cdot \boldsymbol{f}. \tag{2.19}$$

Then, from the caloric equation, the following differential relation can be obtained:

$$dp' = c_0^2 d\rho' + \frac{p_0}{c_v} ds' \tag{2.20}$$

where c_0 is the propagation speed of acoustic signals and c_v is the specific heat at constant volume per unit mass.

Relation 2.20 is a constitutive relation, which can be used together with the 2.18 and the 2.19 to get the in-homogeneous wave equation for pressure perturbations in a quiet and homogeneous medium:

$$\frac{1}{c_0^2} \frac{\partial^2 p'}{\partial t^2} - \nabla^2 p' = \frac{\partial}{\partial t} \left(Q_m + \frac{\gamma - 1}{2} Q_w \right) - \rho_0 \nabla \cdot \boldsymbol{f}. \tag{2.21}$$

While the left-hand side of 2.21 is represented by the d'Alembertian operator applied to the pressure fluctuations, gathering the information about the wavy behaviour of an acoustic signal, the two terms in the right-hand side represent the acoustic field's sources. The first term, containing Q_m and Q_w, is associated to a monopolar behaviour (left of figure 2.1) while the second term, containing $\boldsymbol{f}$, is associated to a dipolar behaviour (center of figure 2.1)[1]. If the right-hand side is null, equation 2.21 describes the propagation of sound sources located on computational domain's boundaries.

[1]A third source term would exist, if the viscous properties of the considered medium were not neglected. This term, containing the influence of shear stresses, would be associated to a quadrupolar behaviour (right of figure 2.1).

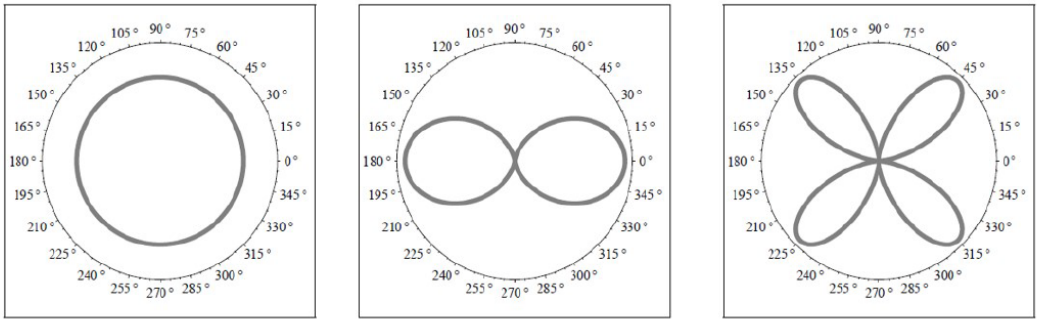

Figure 2.1: Directivity patterns of mopolar, dipolar and quadrupolar source models

2.2.2 Solution with Green's function

An explicit solution to the in-homogeneous wave equation (2.21) can be provided using the method of Green's functions.

Considering a linear differential operator $\mathcal{L}$, applied to a set of distributions defined over a subset Ω within an Euclidean space $\mathbb{R}^n$, a Green's function, indicated as $G = G(\boldsymbol{x}; \boldsymbol{x'})$ and specified for $\boldsymbol{x'} \in \Omega$, is a mathematical tool linked to $\mathcal{L}$ representing any solution of:

$$\mathcal{L}G(\boldsymbol{x}; \boldsymbol{x'}) = \delta(\boldsymbol{x}; \boldsymbol{x'}), \tag{2.22}$$

where δ is the Dirac delta generalized function.

Green's functions can be exploited to solve differential equations of the form $\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x})$, with both $u(\boldsymbol{x})$ and $f(\boldsymbol{x})$ defined in Ω, making the solution explicit as:

$$u(\boldsymbol{x}) = \mathcal{L}^{-1} f(\boldsymbol{x}) = \int_{\Omega} G(\boldsymbol{x}; \boldsymbol{x'}) f(\boldsymbol{x'}) d\boldsymbol{x'}. \tag{2.23}$$

To treat the acoustic wave equation, the Green's function must be defined as $G = G(\boldsymbol{x}, t; \boldsymbol{y}, \tau)$, representing the response, at position $\boldsymbol{x}$ and at time t, to an impulsive point source placed in $\boldsymbol{y}$ that emits a pulse at time τ.

The differential equation, for which G is a solution, has the form of a wave equation with the impulsive source term at right-hand side, formulated as product between two Dirac deltas:

$$\frac{1}{c_0^2} \frac{\partial^2 G}{\partial t^2} - \nabla^2 G = \delta(\boldsymbol{x} - \boldsymbol{y}) \delta(t - \tau). \tag{2.24}$$

As this problem is time-dependant, for $t < \tau$, causality condition requires that $G(\boldsymbol{x}, t; \boldsymbol{y}, \tau) = \frac{\partial G}{\partial t}(\boldsymbol{x}, t; \boldsymbol{y}, \tau) = 0$, if the impulse is the only source of disturbance [6].

The Green's function is further determined by the imposed linear boundary conditions. If the boundary conditions correspond to those of the acoustic field it can be called a *tailored* Green's function [7]. However, as finding it can be difficult, if the considered domain has no walls, the problem can be simplified by computing the so-called *free field* Green's function:

$$G_0(\boldsymbol{x}, t; \boldsymbol{y}, \tau) = \frac{\delta\left(t - \tau - \dfrac{\|\boldsymbol{x} - \boldsymbol{y}\|}{c_0}\right)}{4\pi\|\boldsymbol{x} - \boldsymbol{y}\|}, \tag{2.25}$$

where $\tau = t - \frac{\|\boldsymbol{x}-\boldsymbol{y}\|}{c_0}$ is the *retarded time*. Relation 2.25 describes an outward traveling impulsive wave whose amplitude is inversely proportional to the distance.

Whenever the studied case concerns the sound propagation in a free field, Sommerfeld boundary condition has to be applied:

$$\lim_{r\to+\infty} r\left(\frac{\partial p'}{\partial t} + c_0\frac{\partial p'}{\partial r}\right) = 0. \tag{2.26}$$

Relation 2.26 implies that the involved sound sources radiate waves, scattering their energy to infinity rather than absorbing it.

Equation 2.21 can be rewritten in its integral form, replacing source terms in the right hand side with the general source term q and applying Green's theorem:

$$\begin{aligned} p'(\boldsymbol{x}, t) = &\int_{t_0}^{t}\iiint_V q(\boldsymbol{y}, t)GdVd\tau + \\ &- \int_{t_0}^{t}\iint_S [p'(\boldsymbol{y}, \tau)\nabla G - G\nabla p'(\boldsymbol{y}, \tau)]\cdot\boldsymbol{n}dSd\tau + \\ &- \left[\iiint_V \left[p'(\boldsymbol{y}, \tau)\frac{\partial G}{\partial\tau} - G\frac{\partial p'}{\partial\tau}(\boldsymbol{y}, \tau)\right]dV\right]_{\tau=t_0}, \end{aligned} \tag{2.27}$$

where V is the integration volume, S is the surface enclosing volume V and $\boldsymbol{n}$ is a versor, normal to S and directed towards the outside of V.

In 2.27, the first term on the right side of the equation represents the influence of the sound source. The second term takes into account the effects of boundary conditions, so it is equal to zero if G is a tailored Green's function. The third term represent the effect of the initial conditions, so it vanishes if $t_0 = -\infty$.

If both the simplifying hypotheses are satisfied, 2.27 becomes:

$$p'(\boldsymbol{x}, t) = \int_{t_0}^{t}\iiint_V q(\boldsymbol{y}, t)GdVd\tau. \tag{2.28}$$

2.2.3 Frequency domain formulation

Often, an alternative approach, such as studying acoustic propagation decomposing a signal in different frequencies, turns out to be a convenient tool.

Given the frequency f, the corresponding pressure fluctuation can be defined as:

$$p'(\boldsymbol{x},t) = \bar{p}\ cos(\omega t + \phi) = \bar{p}\ sin(\omega t + \phi') = \Re(\hat{p}(\boldsymbol{x})\ e^{i\omega t}), \tag{2.29}$$

where $\bar{p}$ is the amplitude of pressure fluctuation, $\omega = 2\pi f$ is the pulsation, ϕ and ϕ' are two different values of the initial phase and $\hat{p}$ is the Fourier coefficient for the considered frequency.

If the solution's definition given in relation 2.29 is applied to the wave equation, the in-homogeneous Helmholtz equation can be achieved:

$$\nabla^2\hat{p} + k^2\hat{p} = \hat{q}, \tag{2.30}$$

where $k = \frac{\omega}{c_0} = \frac{2\pi}{\lambda}$ is called wave number ($\lambda = \frac{c_0}{f}$ is the wave length) and $\hat{q}$, the frequency domain version of the source term, comes from the relation $q(\boldsymbol{x},t) = \hat{q}(\boldsymbol{x})\ e^{i\omega t}$.

The in-homogeneous Helmholtz equation, as the in-homogeneous wave equation, can be solved using the formalism of Green's functions. The Green's function to exploit for this purpose must be the solution of the equation:

$$(\nabla^2 + k^2)\hat{G}(\boldsymbol{x} - \boldsymbol{y}) = \hat{\delta}(\boldsymbol{x} - \boldsymbol{y}), \tag{2.31}$$

where $\hat{G}(\boldsymbol{x} - \boldsymbol{y})$ is a frequency-domain Green's function and $\hat{\delta}$ is its corresponding frequency domain source.

Applying the Sommerfeld condition, as the signal propagation is assumed taking place in a free-field environment, the Green's function can be defined as:

$$\hat{G}(\boldsymbol{x} - \boldsymbol{y}) = \frac{e^{ik\|\boldsymbol{x}-\boldsymbol{y}\|}}{4\pi\|\boldsymbol{x} - \boldsymbol{y}\|} \tag{2.32}$$

Then, if the considered source is a monopole, the frequency-domain acoustic pressure can be written as

$$\hat{p}_{mono}(\boldsymbol{x},\boldsymbol{y}) = \hat{a}_{mono}\frac{e^{ik\|\boldsymbol{x}-\boldsymbol{y}\|}}{4\pi\|\boldsymbol{x} - \boldsymbol{y}\|}, \tag{2.33}$$

where $\boldsymbol{x}$ is the listener's position, $\boldsymbol{y}$ is the source's position and $\hat{a}_{mono}$ is the complex amplitude of the monopolar source.

2.3 Tools for the analysis of pressure signals

Pressure fluctuations are the solution of the wave equation, but there are many other effective tools available to describe an acoustic field.

2.3.1 Sound Pressure Level

The SPL (Sound Pressure Level) is one of the most exploited parameters in this thesis and is used to measure the pressure fluctuation intensity of a sound with respect to a reference value. The classic definition of SPL requires the introduction of a new parameter, called the root mean square or effective value, which is necessary as it contains a statistic information about the amplitude of pressure fluctuations. If T is the period of the signal, this parameter is defined as:

$$p'_{rms} = \sqrt{< p'(t)^2 >} = \sqrt{\frac{1}{T}\int_{t_0-\frac{T}{2}}^{t_0+\frac{T}{2}} p'^2(t)dt} \tag{2.34}$$

The "overall" Sound Pressure Level, can then be written as:

$$SPL = 10\log_{10}\left(\frac{p'_{rms}}{p'_{ref}}\right)^2 = 20\log_{10}\left(\frac{p'_{rms}}{p'_{ref}}\right), \tag{2.35}$$

where p'_{ref} is the pressure reference value, usually considered equal to $20\mu Pa$, which is the human hearing threshold.

However, it is often necessary to analyze a signal in the frequency domain, so a frequency-based definition of the SPL must be established:

$$SPL(f) = 10\log_{10}\left(\frac{PSD(f)\Delta f}{(p'_{ref})^2}\right), \tag{2.36}$$

where $PSD(f)$ is the Power Spectral Density referred to the frequency f and Δf is the width of the frequency bins used to compute the PSD.

2.3.2 Power Spectral Density and Welch's method

The PSD (Power Spectral Density) is a continuous function defined in the frequency domain which expresses the power distribution between the frequencies composing the spectrum of the analyzed signal.

In order to compute its value for a given time series $p'(t)$, about a frequency f, the signal's Fourier Transform (FT) has to be computed as:

$$\mathscr{F}[p'(t)](f) = \int_{-\infty}^{+\infty} p'(t)e^{-i\omega t}, \tag{2.37}$$

where $\omega = 2\pi f$.

Then the PSD relative to the corresponding frequency can be computed with the following formula:

$$PSD(f) = \lim_{T\to+\infty} \frac{|\mathscr{F}[p'(t)](f)|^2}{T}, \tag{2.38}$$

where T is the duration of the signal in the time domain.

While many different methods are available to calculate the PSD, in this work the primary tool for its computation is Welch's method, which is a variation of the periodogram method. If a classic TF-based method processes the entire signal in one go, Welch's method divides it into many segments, called windows, which are characterized by a defined width and a certain degree of mutual overlap. Although the amount of width and overlap of the windows are arbitrary, their value determines the frequency resolution of the spectrum and the variance in the PSD estimate. Another feature of this method is the possibility of choosing the shape of the considered windows. This consists in deforming the signal contained in the windows by applying a "weight" to the data depending on their temporal location. The most frequently used window is the Hanning window, having a curved shape with a central maximum. In the end, the information obtained from the periodograms of all windows is averaged, making the result of the PSD calculation with the Welch's method less noisy, although affected by a lower frequency resolution.

2.4 Conventional Beamforming method

This section is used to describe the CB, a method based on the concept of Delay and Sum (D&S) and primary inspiration for the DTB algorithm. With the aim at defining the source location, unknown in advance, this technique is characterized by some peculiar features:

- an array of phased microphones, whose positions are clearly determined;
- a Scanning Grid (SG) constituted by a set of nodes, which are the potential positions of the sound sources.

2.4.1 Time-domain Conventional Beamforming

The first approach to CB is through its time-domain formulation, with the illustration of D&S method.

Considering a linear array of three microphones, arranged with a distance d one from the other, imagine an acoustic wavefront, whose sound source is far enough to consider it the front of a plane wave, hitting all the sensor with a common angle θ, defined between the propagation direction and the straight line where the microphones are positioned (figure 2.2).

As speed of sound c_0 has a finite value, depending on the amplitude of θ, the wavefront stimulates microphone's sensing area at different times, causing signal delay between a microphone and the one next to it:

$$t_{delay} = \frac{d}{c_0} \cos\theta \tag{2.39}$$

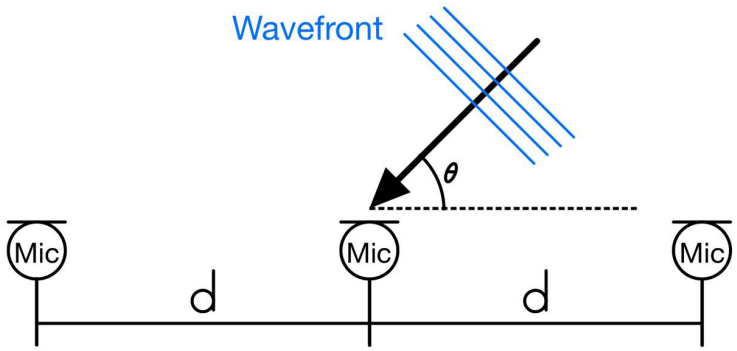

Figure 2.2: Plane wavefront hitting the array of microphones

Only if sound propagation direction is normal to the array plane (i.e. microphones' straight line) $\cos\theta = 0$, so $t_{delay} = 0$ for every couple of microphones.

Considering a single grid point, the first step to apply the C&S method is the estimation of the delay that would affect the recorded signals if the acoustic source was located at the chosen node. This hypothetical delay is applied to the recorded data so that, if it corresponds to the actual delay, the sum of all the shifted signals highlights an effect of constructive interference, showing the growth of fluctuation's amplitude and identifying the grid point as a sound source (figure 2.3). This operation is the numerical equivalent of physically steering the array to expose it perpendicularly to the sound propagation direction, as the resulting sum of microphone's signals would give the same output.

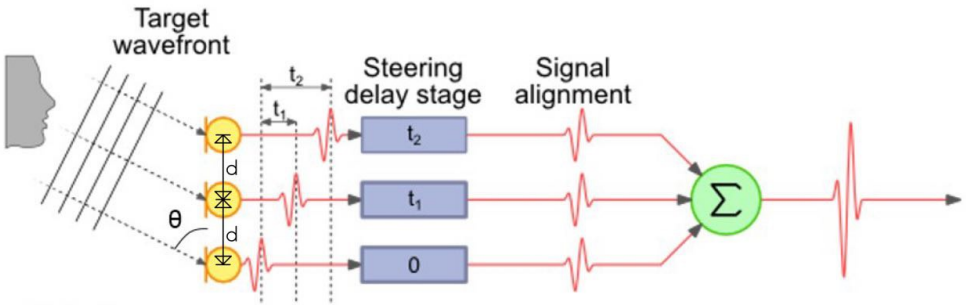

Figure 2.3: Delay and Sum method

If, on the contrary, the evaluated delay time doesn't match the detected one, the output of the method is the sum of out-of-phase signals resulting with a lower amplitude.

A time-domain CB method cycles the same kind of analysis, based on the D&S method, for every point of the scanning grid, exploiting the delay, hidden in recorded data, to establish whether or not each of the grid nodes corresponds to a

sound source [8] [9].

The output of the whole CB method can be plotted as a map, showing the SPL of each point included in the SG, characterizing the acoustic field and showing the position of the sound source.

2.4.2 Frequency-domain Conventional Beamforming

An alternative to the time-domain formulation of the CB method is its frequency-domain version. This latter algorithm must be applied to a proper input, reason why microphones' signals need to be previously translated into frequency domain data through the employment of the FT. Considering the acquisition of one of the N microphones from the array, given the number M of frequencies inspected with the Fourier Transform[2], this new data are characterized by M values representing the corresponding pressure contribution to the fluctuating signal.

Then, it is possible to define a vector containing the pressures contributions recorded by any sensor of the array and associated with a selected frequency:

$$\boldsymbol{p}(f) = \begin{bmatrix} p_1(f) \\ \vdots \\ p_N(f) \end{bmatrix} \in \mathbb{C}^N. \tag{2.40}$$

This class of vectors is essential to compute the Cross Spectral Matrix, $CSM \in \mathbb{C}^{NxNxM}$, whose generic element $CSM_{i,j,k}$ is the cross power spectral density computed between the the elements $p_i(f_k)$ and $p_j(f_k)$, both belonging to vector $\boldsymbol{p}(f_k)$. Usually to ease the problem, an analysis frequency is selected, so that the CSM becomes a square NxN matrix.

After the CSM, it is important to define the normalized steering vector[3]:

$$\boldsymbol{w} = \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|^2}, \tag{2.41}$$

where $\boldsymbol{g}$ is the not-normalized steering vector, a vector that corresponds to the frequency-domain Green's function described to solve the Helmholtz equation:

$$g_{m,n} = \frac{e^{-i\omega\Delta t_{m,n}}}{4\pi\|\boldsymbol{x}_n - \boldsymbol{x}_m\|}, \tag{2.42}$$

where $\boldsymbol{x}_n$ is the position of the n-th microphone on the array and $\Delta t_{m,n}$ is the time gap between the sound emission in position $\boldsymbol{x}_m$ and its reception at $\boldsymbol{x}_n$. The

[2]Considering a discrete FT approach

[3]This formulation of the steering vector is the one predominantly used in this work, but it's not the only available.

output of the CB method can be computed as:

$$CB(\boldsymbol{x_m}) = \boldsymbol{w}_m^* CSM \boldsymbol{w}_m \tag{2.43}$$

2.4.3 Source Power Integration

In order to provide a quantitative result for the acoustic field characterization, easier to compare with classic analysis of acoustic spectra, the concept of SPI (Source Power Integration) is introduced.

The problem of source-maps, the typical output of beamforming methods, lies in the nodes of the SG, associated with SPL values which, frequency by frequency, are specified depending on their position. Therefore, this type of output is not very effective when used to describe the behavior of distributed sources.

A more appropriate way to exploit the DTB results is to select an area of the map, defined ROI (Region Of Integration), which includes all the nodes that contribute most to the sound emission. The contributions of these nodes are subsequently summed to provide a complete characterization of the acoustic field.

Such a result can be achieved by applying the following formula:

$$P_{exp} = \frac{P_{sim}\Sigma_{l=1}^{L} CB(\boldsymbol{x}_l)}{\Sigma_{l=1}^{L}[\boldsymbol{w}_l^*(\boldsymbol{g}_P \boldsymbol{g}_P^*)\boldsymbol{w}_l]} = \frac{P_{sim}\Sigma_{l=1}^{L}(\boldsymbol{w}_l^* CSM \boldsymbol{w_l})}{\Sigma_{l=1}^{L}[\boldsymbol{w}_l^*(\boldsymbol{g}_P \boldsymbol{g}_P^*)\boldsymbol{w}_l]}, \tag{2.44}$$

where P_{sim} is typically taken as one, the subscript l refers to anyone of the L grid points and the subscript P to the center of the ROI [10].

2.5 Noise generation

Ignoring noises produced by tools and devices equipped on a drone as payloads, whose intensities are usually negligible, the most impacting noise generation mechanisms, for a classic quad-copter, are clearly linked to the rotation of its propellers.

The two greatest noise contributions share the aeroacoustic origin, being both of them caused by the presence of unsteady aerodynamic phenomena. One is a narrow-band noise, called tonal, as its emissions are limited to one specific frequency (or a few specific frequencies) and its harmonics. The other is broadband noise, an acoustic emission with variable intensity over a wide range of frequencies.

2.5.1 Tonal noise

In case rotors are characterized by equally spaced blades, tonal noise appears as an emission linked to propellers' rotational velocity. Its origins can be found in steady-state blades' aerodynamics as well as in the periodic variations of blades' aerodynamics over the rotation period [3].

To clarify the relationship between this noise emission and the propeller's rotational speed a parameter, called BPF (Blade Passing Frequency), is defined. Considering a point on the circumference described by the rotation of a propeller, its BPF is the frequency at which a blade pass to that point:

$$BPF = \frac{N_b \Omega}{60}, \tag{2.45}$$

where N_b is the number of blades composing the propeller and Ω is its rotational speed (in rpm). The BPF is the frequency nearby which the tonal noise carries out its main contribution. Thus, thanks to the BPF, it is possible to identify many other frequencies, or harmonics, that gradually influence the acoustic field with reduced intensity as they move further from the BPF (figure 2.4).

Anyways, in its turn, tonal noise is a sum of many noise contributions, caused by different acoustic generation mechanisms, which are explained below.

- **Thickness noise**
  Depending on the shape of the blades, this noise is caused by the air displacement produced on the rotor's plan. It can be modeled as the emission of a monopolar source, but, thanks to its dependence from rotor's shape, thickness noise does not have a big impact on the acoustic field for a small quadcopter.
- **Loading noise**
  Loading noise is generated by the periodic fluctuation of steady and unsteady aerodynamic actions on the blades. It can be modeled as a dipole whose axis is parallel to the propeller's axis.
- **Blade Vortex Interaction**
  BVI is a periodic noise production mechanism in which the vortex produced by a blade of a rotating propeller is invested by the following blade.
- **Blade Wake Interaction**
  BWI noise is the result of the interaction between the turbulent wake produced by a blade and the following blade of the same propeller.

2.5.2 Broadband noise

The main characteristic of the broadband noise is its continuity on the frequency domain. An explanation to this feature can be found in the random fluctuations of blade load. These fluctuations are caused by interactions between the blades (and their boundary layers) and inflow turbulence or the wake generated by the propellers. Other sources are blade-tip vortex formation and interaction with other propellers.

Even if broadband noise affect all the acoustic spectrum, its main contribution is directed to the high frequencies [4].

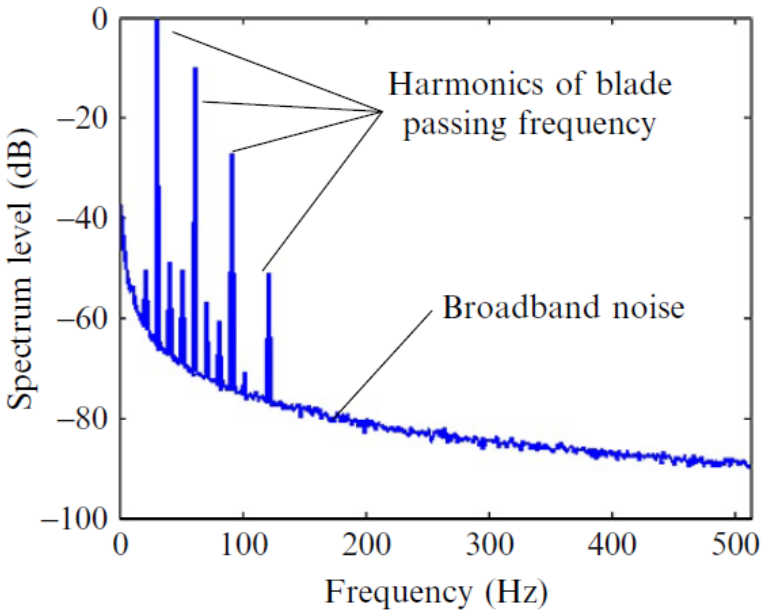

Figure 2.4: The spectrum of rotor noise showing harmonics at blade-passing frequency and broadband noise. [11]

Chapter 3

Software environment

This chapter introduces the code used to prepare, for the DTB method, the data collected by the electrets and the codes used for their processing to get the final beamforming outputs.

3.1 Data preparation

Correctly structured and processed input data are essential to ensure the accurate operation of the DTB algorithm. A code was implemented on MATLAB to prepare the signal, by applying the calibration (explained in chapter 5) and managing the synchronization between the motion-capture system and the microphone array system.

3.1.1 Calibration and synchronization

The script focuses on the data recorded during the experimentations.

In the first part of the process, the calibration is applied to the signals detected by the electrets. This calibration relies on a set of transfer functions (TF) defined in the frequency domain. These functions have previously been discretized and saved within a matrix named TFM, where each column represents a different TF.

The code employs the Fourier Transform on the fluctuating tension signals to compute their frequency domain version, defining for each microphone a vector whose elements can be indicated as $V_i' = V'(f_i)$.

The tension signals can be characterized by a different number of sampling points, within the frequency domain, with respect to those characterizing the TFs. Therefore, the following step is the interpolation of every TF in the TFM, at the frequencies in which the tension signal is defined.

For each microphone, a frequency-domain pressure signal is computed by applying the proper TF to the corresponding tension signal as follows:

$$p'(f) = V'(f)\frac{1}{TF(f)}. \tag{3.1}$$

Then, pressure time-histories are computed from the frequency domain pressure signals, by applying an inverse Fourier Transform.

The second part of the script introduces the data recorded by the motion-capture cameras and synchronizes the trajectories of the markers detected by their system with the audio signal recorded by the array of electrets.

To achieve this purpose, a tool called "sync-stick" was specifically realized by attaching a motion-capture marker on a stick with a wide base. Once that both the systems were recording, before the start of any drones' maneuver, the wide base of the sync-stick was beaten on the ground, producing a deep and loud impulsive noise.

The script detects the the loud noise as a maximum in the pressure-signal's amplitude, then computes the second time derivative of marker's displacement and defines the instant associated with the maximum acceleration, when the sync-stick touches the ground. After that, both set of time-series are cut before the synchronization signal and after the end of the maneuver. The focus is kept on realizing two coherent sets of vectors, considering the difference between microphones' sampling frequency and cameras' sampling rate.

3.2 DTB

The Drone-Tracking Beamforming code is the main tool used to analyze the data gathered during the experimentations in Nivelles and implements the algorithm to be validated. The script, reported in appendix C, is based on the works of Giacomo Gioli Torrione [8] and Riccardo Zamponi [12].

The code takes as input the calibrated microphone's data synchronized with those recorded by the motion-capture system and process them considering many other pieces of information, such as those about the flight environment or about the instrumentation.

At first, all the parameters about the analysis are determined, as the analysis frequency (frequency-domain beamforming algorithms are able to analyze only the contribution of a single frequency at a time) or the number of time-chunks in which the signal must be split. The vectors representing the pressure time-series, from the array of electrets, and drone's trajectory, from the motion-capture system, are divided in segments corresponding to the time chunks. This is, actually, one of the primary features of the DTB method. Also the microphones' disposition inside the

Dougherty array must be fixed, so a matrix is defined as a list whose rows contain the Cartesian components of the positions of each electret.

For each chunk an analogous, but independent, analysis is conducted, repeating for every time-segment the following operations.

- The speed of sound is computed depending on the environmental temperature, and data from the cameras' system are used to compute the drone's velocity components. Then, all these information is used to get an estimation of the aircraft's Mach number, which in turn is used to compute the parameter β, linked to the Doppler effect.

- The SG is created, following the instruction given by the user about its dimensions and the distance between the nodes that compose it. The grid is then positioned at a distance *Gdist* from the antenna, equal to the mean axial distance (during the single time-chunk) between the drone and the center of the array, detected by the cameras' system.

- The CSM is computed as explained in section 2.4.2 using the function *cpsd*, which is based on Welch's method.

- The analysis is then carried out on a limited frequency-band, which has width equal to one third of an octave (or to that of the frequency bins used for the CSM estimation) containing the analysis frequency.

- For each node of the SG, a steering vector $\boldsymbol{g_j}$ is computed and normalized to get the vector $\boldsymbol{w_j}$, then the Beamforming output is calculated with the formula $CB(\boldsymbol{x_j}) = \boldsymbol{w}_j^* \boldsymbol{CSM} \boldsymbol{w}_j$. Then, the resulting source map is plotted and saved.

The steering vector $\boldsymbol{g_j}$, which is the Green's function, can be normalized in many different ways but the result must be characterized by two important features: when the assumed source position coincides with the true source position, the steering vector must determine the maximum value of the beamforming output and, at the same time, the output should be considered as an indicator of the real intensity of the source.

The code has been developed in order to allow the choice between two specific definitions of normalized steering vector.

1. $\boldsymbol{w_j} = \dfrac{\boldsymbol{g_j}}{\|\boldsymbol{g_j}\|}$

2. $\boldsymbol{w_j} = \dfrac{\boldsymbol{g_j}}{\|\boldsymbol{g_j}\|^2}$

The first one is the classic definition of a vector normalization and allows precise localization of the sound source, while the second enhance the beamforming output value computation reducing its dependence from the source distance, which is good for a moving sound source, but loosing precision in the source localization [13].

As this code is mainly exploited to produce source maps, highlighting noise source's position, it is usually set with a single analysis frequency. A second, completely analogous, version of this code was realized enabling the DTB analysis on a wide range of frequencies and implementing the SPI (explained in section 2.4.3), avoiding the of the source maps in order to increase the speed of the algorithm.

Chapter 4

DTB data acquisition

This chapter contains the information about the methodology behind the experimentations conducted to get the data to validate the DTB algorithm. At first an overview on environments and instruments exploited is done, then the sequence of action consisting the acquisition procedure is described.

Two acquisition sessions were carried out about this project and both of them were prepared and organized with the aim at getting the biggest amount possible of DTB data, concerning different kinds of drones' maneuvers and many static monopolar sources at various distances and frequencies (useful as results' control).

4.1 Experimental Environment

4.1.1 ID2MOVE indoor test zone

ID2MOVE indoor test zone (figure 4.1) is a third party facility, in Nivelles (Belgium), that offers a wide environment (a $600m^2$ room area with $8m$ of ceiling height [14]) fitting the needs of any kind of drone's maneuver and avoiding outdoor atmospheric influences on the recorded acoustic signal.

One of the reasons why ID2MOVE test hall was chosen as location for this experimental campaign is that it houses the Qualisys Motion Capture System (QMCS) inside: it enabled the comparison between the results of the DTB analysis and the detected position of special markers, stuck on top of the drones and tracked by the system.

A further description of the instruments composing the QMCS and the ways in which it was used is given in the following sections.

Figure 4.1: ID2MOVE, Nivelles

4.2 Instrumentation

4.2.1 DJI Phantom 3

DJI Phantom 3 PE (figure 4.2) is the biggest between the drones used in this study. It is a popular drone normally used for recreational purposes, equipped with four two-bladed propellers (23.88cm in diameter) mounted on a common horizontal plane with central symmetry. Its weight distribution allows, in normal hovering conditions, the propellers to spin with an equal rotational speed. In table 4.1 Phantom 3 most relevant characteristics can be read [15].

Weight (Battery & Propellers Included)	1280g
Diagonal Size (Propellers Excluded)	350mm
Max Ascent Speed	5m/s
Max Descent Speed	3m/s
Max Speed	16m/s

Table 4.1: Data sheet extract of DJI Phantom 3

4.2.2 DJI Air 2S

DJI Air 2S (figure 4.3) is a small quad-copter drone with foldable propellers (18.29cm in diameter) and structure. This very characteristic of its structure involves that, once the structure gets unfolded, front and back couples of propellers lie on two different horizontal planes. This fact, in turn, brings the drone to be balanced, in a normal hovering condition, only if the two couples of propellers spin at different speeds. In table 4.2 Air 2S most relevant characteristics can be read [16].

Weight	$595g$
Diagonal Size	$302mm$
Max Ascent Speed	$6m/s$
Max Descent Speed	$6m/s$
Max Speed	$19m/s$

Table 4.2: Data sheet extract of DJI Air 2S

Figure 4.2: DJI Phantom 3

Figure 4.3: DJI Air 2S

4.2.3 Electret microphones

Knowles electrets model FG-23329-P07 (figure 4.4) are the chosen microphones to detect the pressure signals. They are quite inexpensive and, with their small circular sensing area (2.57 mm in diameter), can be closely arranged to form the small central microphone ring of the array. As the sensing areas are connected to 1 m long wires, every electret is inserted into a 13 cm long brass tube and flush mounted to one of its extremities. In order to protect the lateral portion of the sensing area and cover the connection between the sensor and its wire, the brass tubes are fixed in that position with a heat-shrinking tube (figure 4.5). This

configuration also give the electrets structural stiffness, which is necessary to keep them clamped into the holes on the antenna structure (see 4.2.5).

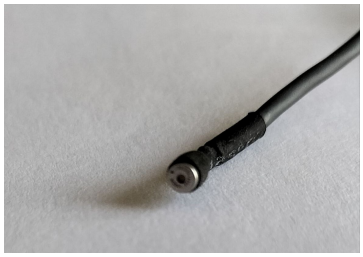

Figure 4.4: FG-23329-P07 Knowles electret

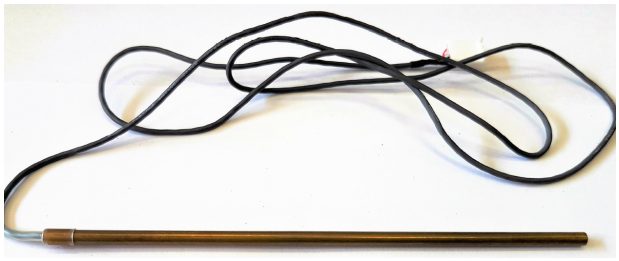

Figure 4.5: Electret fixed to its brass tube

4.2.4 Electrets' amplification system

Electrets produce a low output signal, requiring the use of an amplification system (figure 4.6) consisting of 2 amplifiers, developed at the VKI, and a single power supply box. Each amplifier can be connected to up to 32 microphones at a time and includes a toggle selector that enables users to choose between gain values of 100 or 1000. The power supply box, instead, can be simultaneously connected to up to 4 amplifiers. Throughout both the experimental sessions each amplifiers were set at a gain value of 100.

Figure 4.6: Electrets' amplification system with cables

4.2.5 Antenna structure

The expression "antenna structure" is used, in this thesis, with respect to the white 3D-printed resin spiral shape (figure 4.7), supporting the electrets during DTB experimentations. This structure, with a $1.5m$ diameter, is held (by a metal pole) with its center at a constant height of $125cm$ and is characterised by 64 holes, in which all the electrets (with their brass tubes) can be clamped in. Holes' distribution, on the spiral antenna structure, follows the pattern of a Dougherty array (figure 4.8) [17], allowing the reduction of spatial aliasing with respect to other simpler configurations [18].

The pole, holding the antenna in position, is also employed to support the cables connecting electrets and amplifiers (figure 4.9), in order to avoid the electrets to be pulled away from their position by the weight of the cables.

Figure 4.7: Resin structure

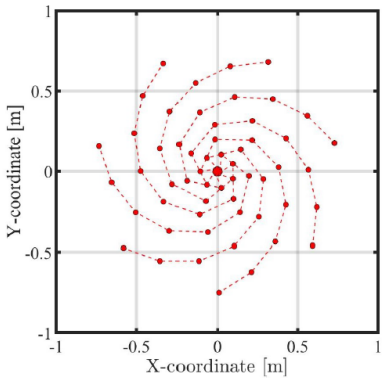

Figure 4.8: Dougherty array, rear view disposition

Figure 4.9: Complete antenna

4.2.6 PXI recording system

The recording system for the pressure signal employed in this study was a National Instruments PXI, composed by a PXIe-1075 chassis and a PXI-8105 system controller, a high performance processor with 8 acquisition slots, numbered from 2 to 9. Each acquisition slot contains 8 pin sockets, numbered from 0 to 7, so that it is possible the simultaneous connection with all the 64 SMB cables to the phased electrets (figure 4.10).

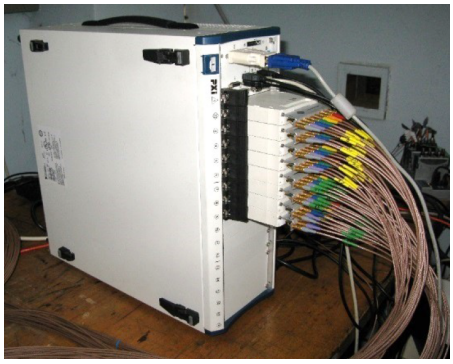

Figure 4.10: PXI recording system, DTB configuration

4.2.7 Connection cables

Two types of cables were employed as connections between the electrets and the recording system: one to connect the electrets to the amplification system and the other to connect the amplification system to the PXI recording system (figures 4.6 and 4.9).

About the first set, since every electret has only a 1-meter-long wire, a white extension cable was connected to each sensor in order to reach the proper amplifier, even from a far location (e.g. antenna's position). About the second set, both the amplifiers were connected to the PXI recording system via 10-meters-long SMB cables. Every cable is numbered in order to let the operator figure out every electret connection to the amplifier and then to the recording system.

4.2.8 Qualisys motion capture system

With twelve available Arqus A5 high-speed motion capture cameras[1], the QMCS allows to record with a frame rate up to $700fps$, an image resolution of $5MP$ and a maximum capture distance of $26m$ [19]. The whole set of cameras can be managed with the software Qualisys Track Manager (QTM), on a conventional computer, that makes possible to record and process the data detected (figure 4.11).

To observe the movements of any kind of object, the QMCS needs the employment of a particular kind of reflective markers (figure 4.12).

To ensure a successful tracking, these need to be affixed onto the surfaces of the objects that are to be monitored and sufficiently distanced. Moreover, these objects must be constantly kept within cameras' sight area.

[1]Only eleven cameras were active during the experimentations, with negligible consequences on results' quality

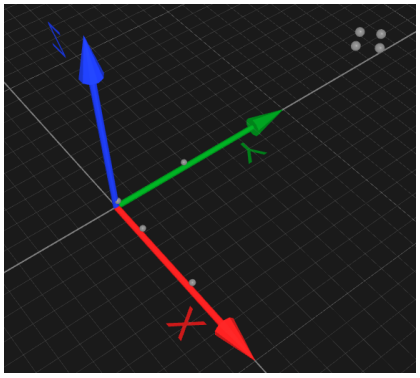

Figure 4.11: Markers in QTM environment

Figure 4.12: Markers on DJI Phantom 3

4.2.9 Sync-stick

The sync-stick (in figure 4.13) is a tool built with three elements:

- a plastic stabilizer weight with a large flat base and a hole at the top;
- a wooden stick with the lower extremity wedged into the hole of the plastic stabilizer weight;
- a reflective marker stuck at the upper, unconstrained, extremity of the wooden stick.

The employment of this tool is explained in section 4.4, however its purpose is completely analogous to that of a typical movie clapperboard: the synchronization of "audio" (from electrets) and "video" (from cameras) data from the same recording.

Figure 4.13: Sync-stick

4.2.10 Agilent 33120A

The Agilent 33120A signal generator (figure 4.14) was used to generate different kind of signals, used as reference for the DTB analysis.

The device allows to select between a constant frequency and a band sweeping signal generation. For the latter mode, it offers the possibility to chose between a linear and logarithmic sweeping, as well as to set initial and ending frequencies of the band.

It also allows to select a basic intensity of the signal generated: throughout all the experimentations, this setting was kept at a value of $100mV/Pa$ but was not precisely exploited, as a further amplifier was connected between the signal generator and the loudspeaker.

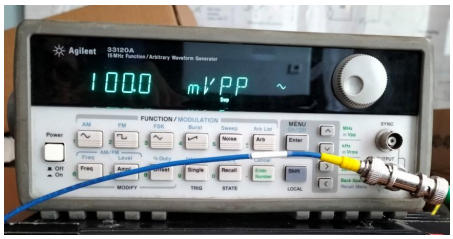

Figure 4.14: Agilent 33120A

4.2.11 JBL Model 6230 Power Amplifier

The JBL amplifier was used to amplify the signal generated by the Agilent 33120A, before delivering it to a loudspeaker. Only channel B was ever employed, with a setting about the attenuation level equal to $28dB$.

Figure 4.15: JBL amplifier

4.2.12 Monopolar source tube

The monopolar source tube is a device, that generates a wide selection of signals, to be used as a control about the coherence between the results of the DTB and

the motion capture system, or as a reference to study SPL's variations with the distance from the electrets' array.

It was built connecting a loudspeaker to the extremity of a rubber tube, leaving the other end open to the external environment and, so, making its small opening coincident to the actual position from where the sound was spread. The loudspeaker, in turn, was connected to the JBL Model 6230 Power Amplifier and through it to the Agilent 33210A signal generator, previously described.

4.3 Set up

The set up for this experiment was realized with the aim at the storage of a large amount of data, from two different acquisition systems working simultaneously, to store them at a later time in the same portable hard drive (figure 4.16).

The first acquisition system is the one designated to the detection and recording of acoustic signal. It is obtained mounting 64 electrets on the antenna structure, and connecting them to their amplification system and then to the PXI recording system. The second acquisition system is the QMCS, previously described.

The same set up was applied to extract data about two different kinds of sound sources: monopolar sources and drones.

4.4 Acquisition procedure

As the data acquisition was a fundamental part in the development of this project, it was carefully prepared as a sequence of actions to repeat at any take.

Even before the recording sessions, it was clear that one of the main issues, with a similar set up, was the synchronization of the time histories stored in two different types of data file, acquired using two different sample frequencies and without the possibility to precisely control the record starting instant on both the systems at the same time. This is the reason why the sync-stick had been built and deployed at the beginning of every acquisition.

The first item on the experimental check-list was the definition of a common name for both the acoustic and the motion-capture data files, to easily couple them during the data processing, as well as a proper and equal duration for the acquisitions of the two systems. It was customary to launch the collection of pressure data first and then start the acquisition with the cameras.

Thereafter, as soon as the sound source (the monopolar source or one of the drones) was ready for the take, the sync-stick's lower extremity was beaten on the ground, spreading a strong impulsive signal, clearly recognizable in the audio sample, thanks to the instantaneous pressure level peak, as well as in the cam-data

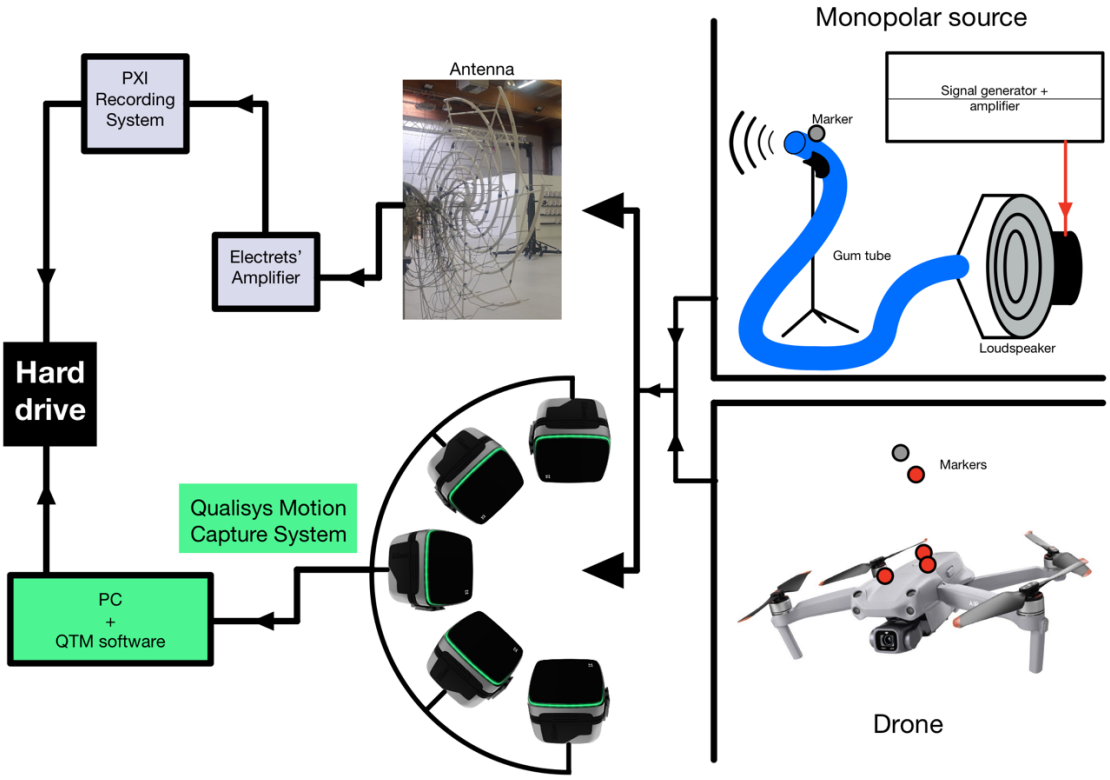

Figure 4.16: Set up for DTB acquisitions

file where it can be found as a peak in vertical acceleration of the sync-stick's marker.

During the recording phase, apart from the background noise and the contributions of the analyzed source, there were no noises until the end of the take. Besides, as the area covered by the cameras sight was limited, only a part of the test hall have been used for the motion-capture operations.

Chapter 5

Electrets' calibration

DTB is a method that needs a large amount of sensors to build a Dougherty array, requiring to place them in the center close to each other, so electrets are particularly well suited for it, being these microphones cheap and small. However, they require frequent calibrations as their response to the same signal can vary over a relatively short amount of time.

This chapter presents a specific calibration procedure which is applied to the whole set of electrets. At first, there is an explanation about the procedure, then the environments and instruments utilized to gather the necessary data are listed and described, presenting the acquisition set-up. Thereafter, the code employed to implement the calibration is explained and, at last, its results are commented and showed with some plot.

5.1 Procedure description

In this project, microphones' calibration is performed by applying a proper Transfer Function (TF) to each of the signals recorded by the 64 electrets during the experimental sessions in Nivelles, in order to improve the quality of the DTB results. These TFs also makes the conversion of tension signals into the corresponding pressure fluctuations.

Here, the calibration procedure is explained by dividing its exposition in two parts: the first one is a general dissertation about the method, the second is about the solution to a resonance issue that affect the TFs at some frequencies.

5.1.1 Method explanation

A TF is a function, designed as link between an input signal and an output signal, that creates a model for the estimation of every possible output, given any possible

value of the input. With this in mind, it is possible to consider a link between a certain reference pressure signal, recorded by a reference microphone, and the signal recorded by an electret to calibrate: in this case, the goal is the estimation of a TF, characterised by the reference ("ref") signal as input and by the signal of the electret ("ele") as output.

The TF can be defined, in the frequency domain, as a complex function with variable f:

$$TF_{ref \to ele}(f) = \frac{\mathscr{F}[V'_{ele}(t)]}{\mathscr{F}[p'_{ref}(t)]}, \tag{5.1}$$

where $\mathscr{F}[V'_{ele}(t)]$ and $\mathscr{F}[p'_{ref}(t)]$ are the Fourier transforms of electrets' tension signal and reference pressure signal respectively.

In order to compute a TF with such a purpose, it is essential to measure the acoustic field under identical conditions (i.e. the acoustic field generated by the same signal, the sensors located in the same position of the same environment) with both the sensors used to acquire input and output signals. Part of this "ideal" requirement can be respected realizing a calibrator, a cave tool, to which a microphone can be mounted. Its cavity has a loudspeaker, as top wall, which is employable for the generation of a specific calibration signal. However, another part of the same requirement cannot be fulfilled: even if the cavity can confine the signal propagation volume and avoid the external acoustic influences, to record the calibration signal in identical conditions, reference microphone and electret should record the signal in synchrony being physically superimposed. As this is not possible, the calibration procedure is split in two different and non-simultaneous steps: one for the reference sensor and one for the electret (figure 5.1).

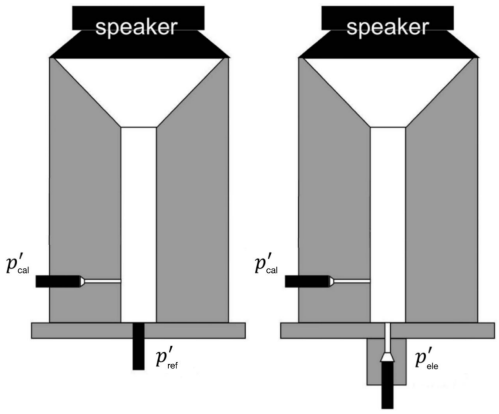

Figure 5.1: Calibration procedure splitting in 2 steps

If the procedure splitting is a drawback to overcome, its solution is linking of the two non-simultaneous steps through a couple of pressure signals recorded by a

third sensor, denoted as calibrator microphone ("cal"), during both the steps. The first "cal" signal is recorded in synchrony with the "ref" signal while the second is recorded in synchrony with the "ele" signal. The two couples of synchronized signals are then used to estimate two different TFs:

$$\begin{aligned} TF_{ref\to cal_1}(f) &= \frac{\mathscr{F}[p'_{cal_1}(t)]}{\mathscr{F}[p'_{ref}(t)]}, \\ TF_{cal_2\to ele}(f) &= \frac{\mathscr{F}[V'_{ele}(t)]}{\mathscr{F}[p'_{cal_2}(t)]} \end{aligned} \tag{5.2}$$

At this point, the concern about the two "cal" signals must be about the equality of their spectra, as this analysis involves their Fourier transforms only. This means that, if the reference acoustic signal produced by the loudspeaker is periodic and an equal number of periods are recorded by all the microphones, during the two non-simultaneous calibration steps, $\mathscr{F}[p'_{cal_1}(t)]$ and $\mathscr{F}[p'_{cal_2}(t)]$ are equals and the link between the two steps actually holds. Indeed, thanks to the linearity property of the TFs, using the two relations in 5.2, the 5.1 can be rewritten as:

$$\begin{aligned} TF_{ref\to ele}(f) &= \frac{\mathscr{F}[p'_{ele}]}{\mathscr{F}[p'_{ref}]} = \\ &= \underbrace{\frac{\cancel{\mathscr{F}[p'_{cal_1}]}}{\mathscr{F}[p'_{ref}]}}_{TF_{ref\to cal_1}} \underbrace{\frac{\mathscr{F}[p'_{ele}]}{\cancel{\mathscr{F}[p'_{cal_2}]}}}_{TF_{cal_2\to ele}} = TF_{ref\to cal_1}\cdot TF_{cal_2\to ele}. \end{aligned} \tag{5.3}$$

As it is a complex function, a TF can be completely defined knowing its amplitude and phase. In figure 5.2 two partial TFs, corresponding to first and second steps of the calibration procedure, and their composition are showed.

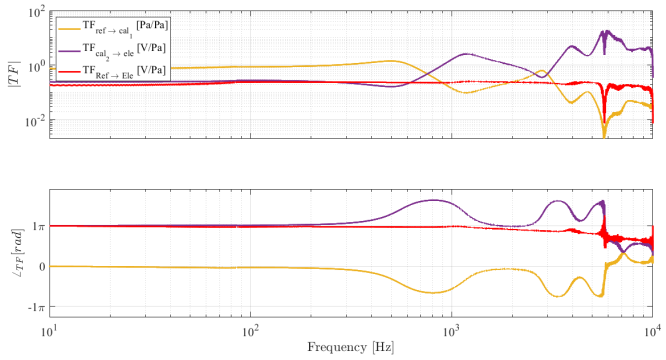

Figure 5.2: TFs from two steps of the calibration procedure

5.1.2 Reflections in the calibrator

It is evident from the image in figure number 5.2 that a TF, defined following the method previously described, presents unexpected irregularities and peaks.

The reasons for this behavior are identified in the presence of resonances in the calibrator system, with influences from "cal" microphone's position, on the side wall of the calibrator.

To solve this problem a second "cal" microphone has been installed on the calibrator's wall at a different distance from the sensor to calibrate (figure 5.3). Hence, a completely analogous calibration procedure has been performed, in order to get an alternative result using the new "cal" signal as bridge between the two calibration steps. The result is a TF similar to the original one, with irregularities and peaks at different frequencies.

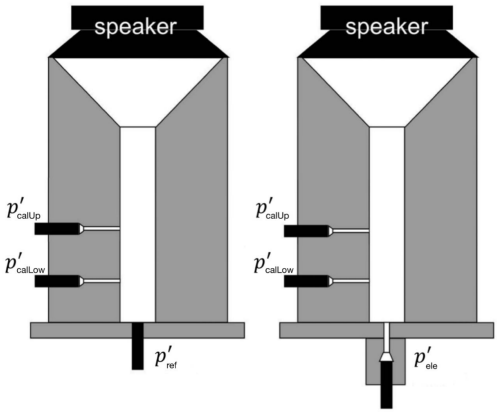

Figure 5.3: Calibrator with 2 lateral B&Ks

Excluding the irregularities of both the "final" TFs, a third "merged" TF has been manually obtained and used to achieve a first approach to electret's calibration. However, in order to further improve the data quality, an interpolation of the most regular segments of this latter TF was performed (figure 5.4).

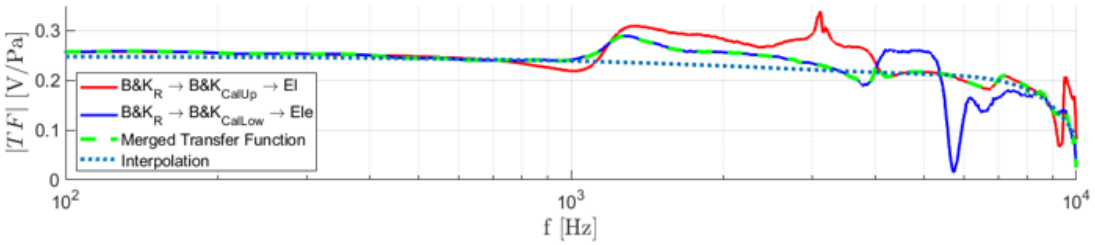

Figure 5.4: Interpolation of the TFs

The function thus obtained is capable of converting the data collected by the "ref" microphone into those collected by the electret. The interesting thing about the TF is that, thanks to linearity again, it is sufficient to compute its reciprocal function to get as output the "ref" pressures using as input the electret's tensions, which is the needed operation:

$$TF_{ele \to ref} = \frac{1}{TF_{ref \to ele}} \tag{5.4}$$

As two experimental data collection sessions were conducted, two respective electrets' calibrations were carried out at close times to the experiments, in order to ensure the best possible data quality.

5.2 Experimental environments

In this short section the laboratories used to carry out the calibrations' acquisitions are described.

5.2.1 Laboratory of Alcove anechoic chamber

Alcove is one of the VKI's anechoic chambers, placed in the basements of one of the institute's buildings. Its laboratory offers a quite environment and a large desk, matching the necessities for electrets calibration procedure. This is the location where signal acquisitions for the first electrets' calibrations were performed.

5.2.2 Laboratory of Jafar anechoic chamber

Another of VKI's facilities, Jafar, is an anechoic chamber with a wide laboratory designed as an open-plan environment. With no walls between Jafar's laboratory and many other experimental stations, noise from the closest facilities was one of the main concerns during the acquisitions for the second round of electrets' calibrations. These acquisitions were performed under optimal recording conditions, ensuring minimal disturbance from nearby machines or instruments.

5.3 Instrumentation

5.3.1 Electret microphones and amplification system

The 64 electret microphones and their amplification system, composed by two specially-made amplifiers, are the same used for the experimentations and described in sections 4.2.3 and 4.2.4.

5.3.2 Brüel & Kjær microphones

Brüel & Kjær type 4938-A-011 - 1/4 inch pressure field microphones (figure 5.5) are sensors with a frequency response ($\pm 2dB$) ranging from $4Hz$ to $70kHz$ and dynamic range between $42dB$ and $172dB$. These microphones are chosen as reference sensors for electrets' calibrations because of their higher accuracy, compared to the latter. Moreover, they do not need frequent calibrations as electrets, but rather sporadic sensitivity computations. Three of these sensors are used for the calibration procedure, assuming the roles of "ref" and "cal" microphones.

Figure 5.5: 4938-A-011 - 1/4 inch pressure field B&K microphone

5.3.3 B&K Nexus Conditioning Amplifier

B&K microphones require a proper amplifier: the Nexus Conditioning Amplifier (figure 5.6). This device is characterized by 4 input/output channels and, for each of them, offers the possibility to select the gain from a preset list.

Throughout the calibrations, only channels 1, 2 and 3 were used to connect the microphones, while channel 4 remained unplugged. The chosen gain for those operations had been the preset level $100mV/Pa$, in order to avoid B&Ks' signal saturation.

5.3.4 PXI recording system

PXI recording system is described in section 4.2.6. However, for the calibration procedures, only channels 0, 1, 2 and 3 of the slot number 2 were used (figure 5.7).

5.3.5 Connection cables

Both the bundles of wires described in section 4.2.7 were used to connect the electrets to their amplifiers and the latter to the acquisition system.

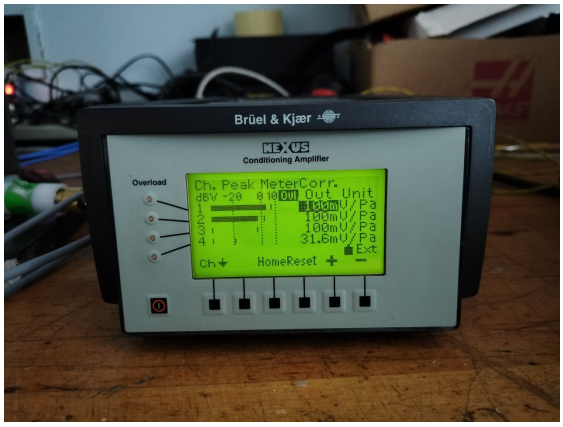

Figure 5.6: B&K Nexus Conditioning Amplifier

However, a third type of wire, the SMB to BNC cable, was employed during the calibration to connect every single B&K channel's output, on the Nexus Conditioning Amplifier, to its corresponding pin socket, on the PXI recording system.

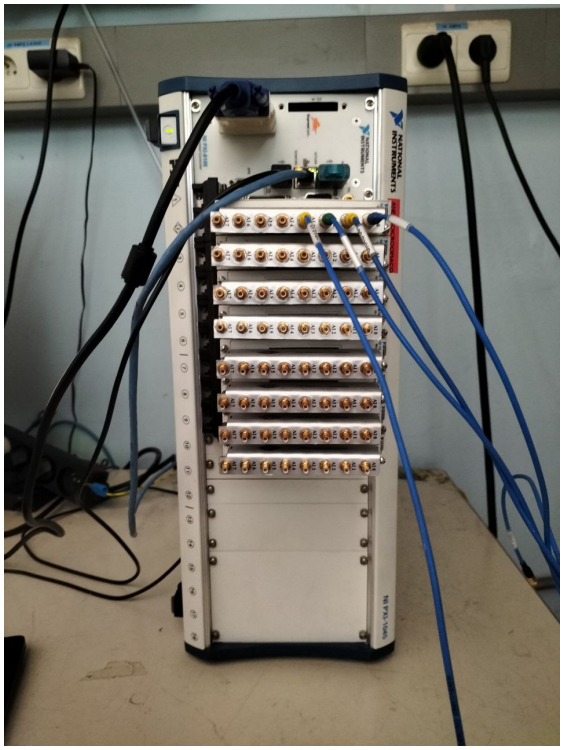

Figure 5.7: PXI recording system: calibration configuration

5.3.6 Signal generation system

This system was composed by the signal generator Agilent 33120A, described in section 4.2.10 and by the JBL Model 6230 Power Amplifier, described in section 4.2.11.

5.3.7 Calibrator

The calibrator (figure 5.8) is a cylindrical metal block with a cavity axially developed, in vertical direction, with a loudspeaker attached at the top.

Its structure allows the propagation, inside the cavity, of a plain wave generated by the loudspeaker and directed towards the bottom.

At the base, the calibrator is open and its cavity exposed to the outer environment, but a damping leather membrane is glued to its bottom wall, in order to reduce vibration transmission when it is placed on a calibration plate. Two B&K microphones can be attached to its lateral walls, thanks to specific plastic stops, in which the microphones can get clamped.

Figure 5.8: Calibrator

5.3.8 Calibration plates

The calibration plates are metal plates with small holes in the center, which are the places where an electret or the reference B&K microphone can be clamped in during the calibration procedure. Therefore, the whole assembly built by plate and microphone becomes the bottom boundary of the calibrator's cavity.

5.4 Set up

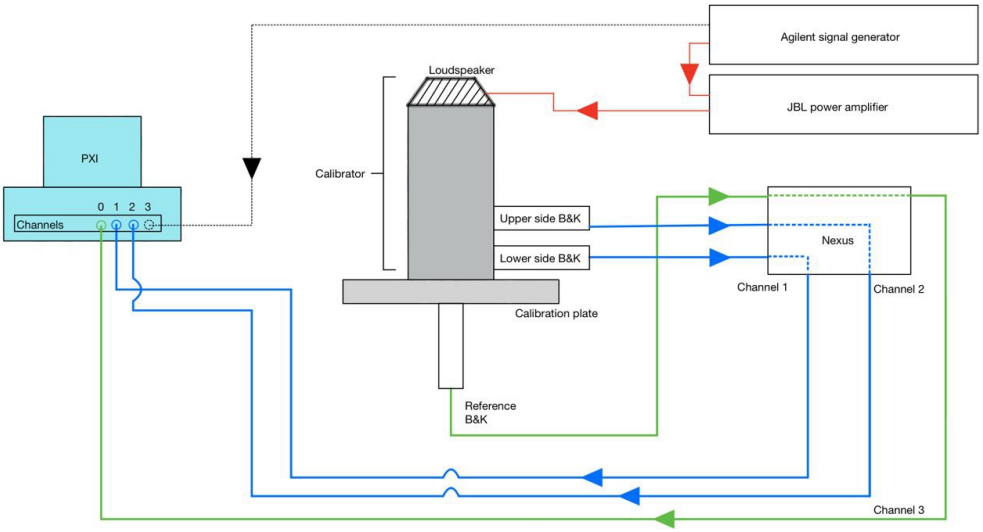

Figure 5.9: Set up for the first calibration step

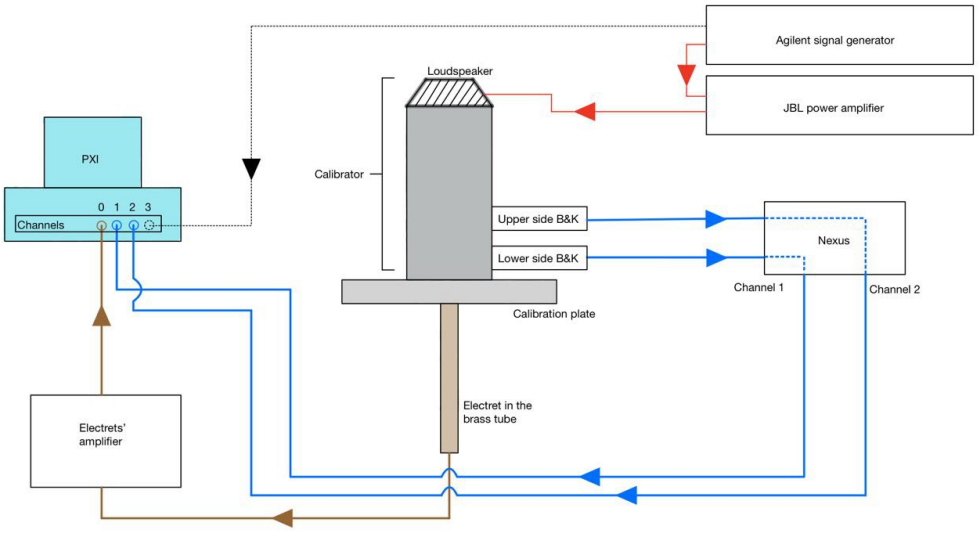

Figure 5.10: Set up for the second calibration step

Materials and instruments aforementioned are used in both the calibration sessions performed. All the electronic instruments have been connected to the ground in order to avoid electrically induced alterations of the signals recorded.

During this procedure, only 4 of the 64 PXI's available channels were used at the same time. Channel 0 was plugged by the connector to the "ref" B&K during the first step of the calibration and to one of the electrets during the second. Channel 1 and 2 have always been kept linked to the lateral B&Ks, just like the channel 3 has always been connected to Agilent signal generator (in case of necessity of a different reference signal).

Just to specify, the "cal" B&Ks and, only during the first calibration step, the reference ("ref") B&K were connected to the PXI through the Nexus Conditioning Amplifier.

The Agilent signal generator was also connected to the JBL power amplifier and then to a loudspeaker on top of the calibrator, to generate the acoustic field inside the cavity of the calibrator itself.

Even if, with this procedure, only a single electret could be calibrated at a time, to minimize errors and variations, every electret have been kept all the time linked to their own amplifier connector, with the aim of getting the best possible calibration.

The lower extremity of the calibrator's cavity is the position where the "ref" B&K and the electret microphones were put during the acquisitions. To do that, the microphones were flush mounted on a metal plate, so that microphone and plate together could build the cavity's bottom.

5.5 Acquisitions

After the set-up preparation, one step at a time, each of the microphone to be clamped into the calibration plate was prepared, applying the correct connection to the acquisitions system. As the first acquisition step is characterized by a set of signals which is identical for all the 64 electrets, it was performed once for all. In this case, the "ref" B&K was connected to the third channel of the Nexus amplifier and then its output cable was pinned into the channel "0" of the acquisition card in the second slot of the PXI. Each of the 64 second steps of the calibration was performed separately, with the same procedure of the first.

The Agilent 33120A signal generator was set to generate a sinusoidal linear sweeping $100mV/P$ signal, ranging from $100Hz$ to $10kHz$, with a sweeping time of $1s$. This signal was then amplified with the JBL model 6230 power amplifier, whose attenuation was set at level $28dB$.

Each acquisition had a duration of $30s$, containing "30" sweeping cycles, which is a trade off between the long-last duration of the whole acquisition session and the number of cycles recorded in a single sample, as a longer sample is statistically more valuable.

5.6 Calibration script

Once the acquisitions of the calibration data are completed, a Matlab code (see Appendix B) is used to compute the TFs to apply on experimental data.

The calibration script estimates a set of transfer function, one for each electret microphone in the array, starting from a collection of input and output tension or pressure time series, specifically recorded for the calibration.

The script exploits Matlab's function *tfestimate*, which automatically provides a vector containing a discretized transfer function and is based on Welch's method. Thus, in addition to the input and output signals, this function requires the usual Welch's method input parameters, about windows and frequencies' definitions:

- *window*: the number of elements contained in each window;
- *noverlap*: the number of overlapping elements between each couple of windows;
- *nfft*: the number of frequency-domain's sampling points;
- *fs*: the sampling frequency of the two processed signals.

As previously explained, for every single microphone, two couples of "partial" TFs are computed with *tfestimate*, using combinations of four different input signals. Then, each of the two couples of "partial" TFs is combined and separately used to obtain a different "final" TF.

Considering, as example, a generic couple of "partial" TF vectors, thanks to their linearity, the "final" TF is computed by multiplying their values together:

$$TF_i^{final} = TF_i^{one} \cdot TF_i^{two}, \tag{5.5}$$

where subscript i refers to anyone of the N_f elements composing the vectors. N_f is the number of sampling points where the TFs are defined in the frequency domain and is parented to the parameter *nfft*.

Even though the two obtained "final" TFs are analogous, they differ in some values associated to some critical frequency ranges. The reasons for this behaviours are linked to the presence of resonances into the calibrator's cavity and to the way the signals' acquisitions are realized. To overcome this issue, the script builds a "merged" TF selecting, segment after segment, the best-behaving "final" TF between the two and then creates an interpolation to cut out the residual irregularities.

The result of this part of the script is a matrix denoted as TFM, whose columns are vectors containing the interpolated TF's complex values of each microphone, defined in the frequency domain:

$$TFM = \begin{bmatrix} \begin{pmatrix} TF^{interp}_{i=1,j=1} \\ \vdots \\ TF^{interp}_{i=N_f,j=1} \end{pmatrix} & \cdots & \begin{pmatrix} TF^{interp}_{i=1,j=N_{mic}} \\ \vdots \\ TF^{interp}_{i=N_f,j=N_{mic}} \end{pmatrix} \end{bmatrix} \in \mathbb{C}^{N_f \times N_{mic}}, \tag{5.6}$$

where j refers to anyone of the N_{mic} microphones composing the array.

The TFM is saved and used in the script for the data preparation.

As the obtained TF is useful to compute the electret tension signal starting from the acoustic signal that excites the microphone ($ref \rightarrow ele$), we need to apply the "reverse" TF to the recorded tension signal to get back to the pressure fluctuation, which is obtained computing its reciprocal function.

5.7 Application to recorded data

The so far obtained inverted, merged final-TF is still defined in the frequency domain, while the tension sample we want to transform in a pressure signal is a time history of data.

To apply the TF and accomplish the calibration it is necessary to bring the pressure signal in the frequency domain.

$$V'_{ele}(t) \rightarrow V'_{ele}(f)$$

In this case a discrete Fourier Transform has been computed.

It is now possible to apply the TF to the signal:

$$p'_{ele}(f) = V'_{ele}(f) \cdot TF_{ele \rightarrow ref}$$

The following step would be the return to the time domain with the pressure signal, using an inverse Fourier Transform:

$$p'_{ele}(f) \rightarrow p'_{ele}(t)$$

Ideally, thanks to the TF, the result should be a pressure time history identical to the one recorded with the "ref" B&K.

The real result can be plotted and compared to the acquisition of the reference microphone (figure 5.11), showing many negligible differences, as the result is acceptable

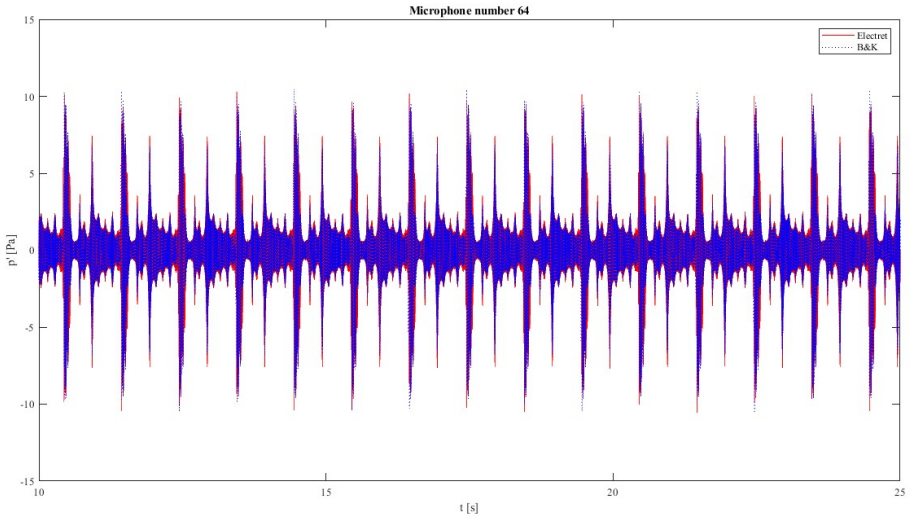

Figure 5.11: Electret and "ref" B&K pressure signal comparison

Chapter 6

Results

In this chapter, calibrated data are analyzed and the results are plotted in spectra, spectrograms or source maps, in order to show the typical outputs of the method and its capabilities in drone tracking. Furthermore, a comparison is made between the spectra of the signals recorded from the central microphone of the phased array and the results of the SPI.

6.1 Tracking of monopolar sources

The first type of analyzed data are those about the monopolar source tube signals. This kind of acquisition was thought as a basic control on the method: once the pressure fluctuations come out of the open extremity of the monopolar source tube, whose opening is narrow enough to be considered as a point source, the acoustic signal propagates as a sphere. Thus it is particularly easy finding the correct position of the tube's opening in the scanning grid.

In table 6.1 the test-matrix for the monopolar source experiments is shown. These experiments are studied in couples in order to show the behaviour of the results considering variations of the analysis frequency and in the distance between the antenna and the source.

	Distance [m]	Duration [s]	Type	Frequency [Hz]
m1	2	6	Constant Frequency	500
m2	2	6	Constant Frequency	4500
m3	1	15	Logarithmic Sweeping	100 to 10000
m4	3	15	Logarithmic Sweeping	100 to 10000

Table 6.1: Monopolar source test-matrix

The first two experimental data-set are characterized by different frequencies,

but keep a constant axial distance of $2m$ between the antenna and the tube's open extremity. The third-one and the fourth-one are the recordings of logarithmic sweeping signals, set with a constant sweeping range but acquired with a variable axial distance.

6.1.1 Monopolar sources: spectra

Experiment m1 considers a signal whose spectrum is plotted in figure 6.1 with the associated background noise. It shows as expected a peak at the fundamental frequency, which is $f = 500Hz$, followed by many other decreasing peaks corresponding to the harmonics, at frequencies defined by $f_H(n) = 500 \cdot n$, where n is a positive integer number.

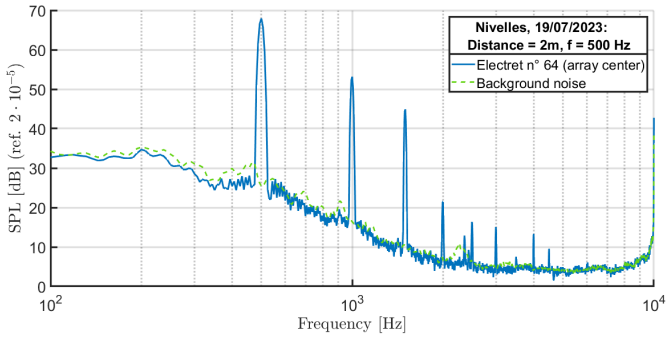

Figure 6.1: PSD of monopolar source experiment m1

The spectrum for the experiment m2 is plotted in figure 6.2 and shows an analogous situation, but with less peaks, as the fundamental harmonic is characterized by a frequency $f = 4500$ and the second, and only visible, harmonic is at $f_H(n = 2) = 4500 \cdot 2 = 9000Hz$.

Both the takes are characterize by a signal that respect the background noise, but show some fluctuation. This is due to the shortness of the analyzed signals, which are good enough to be considered as reference recording, but required to cut out their initial segment which showed spoiling traces of the sync-stick noise.

Spectra associated to experiments m3 and m4 are plotted together in figure 6.3. Their comparisons with the background noise spectrum make clear that the signals are not focused on single frequencies, spreading an increasing contribution for the PSD over the entire band affected by the generation of the signals.

Since both the blue and the red lines are representative of identically generated signals, it is not surprising that they are very similar in shape, which is dependent on the logarithmic sweeping mode and on the efficiency of the loudspeaker in producing the signal at a certain frequency. However, the different axial distance between

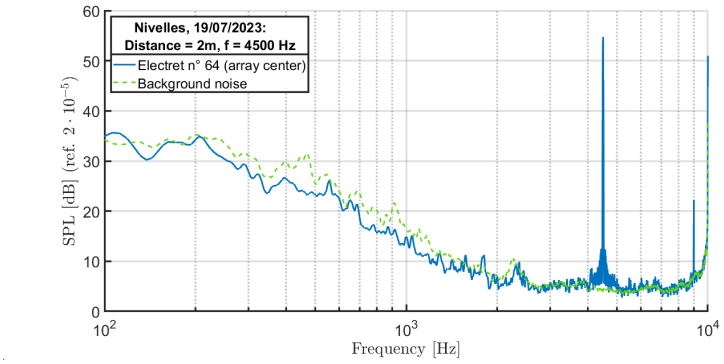

Figure 6.2: PSD of monopolar source experiment m2

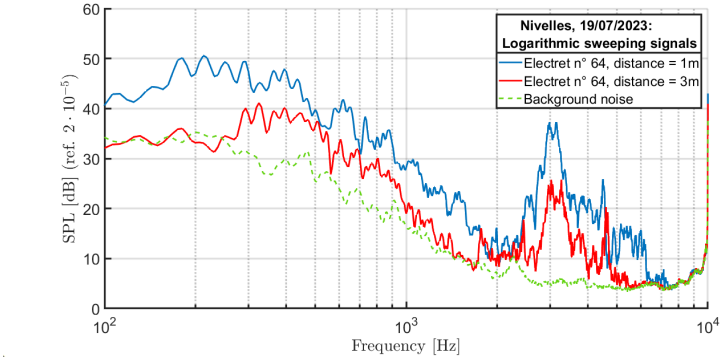

Figure 6.3: PSDs of monopolar source experiments m3 and m4

the antenna and the monopolar source is clearly manifested by the difference in intensity of the signal perceived by the sensors.

6.1.2 Monopolar sources: source-maps

The analysis for this type of signals, from a static source, was conducted by keeping the pressure time series recorded by the microphones not divided into chunks.

All the source-maps for the four considered recordings identify two salient points:

- the position where the maximum value of the beamforming algorithm's output is located, which is indicated with a black cross;
- the time-average of the position occupied by the tracking marker, detected by the motion-capture cameras, which is indicated by a red circle.

Every map also features a coordinate system such that the origin corresponds to the projection of the microphone array's center onto the scanning grid plane. The

word "ground" is written above a dashed line, which identifies the ground level.

About experiments m1 and m2, the resulting source maps are showed in figures 6.4 and 6.5.

With SGs characterized by a grid-spacing of $1mm$, along both the horizontal and the vertical dimensions, the distances between the black cross and the red circle are respectively of $65mm$ for the experiment m1 and $12mm$ for the experiment m2. However, even without the maximum output location it would be clear, from a comparison between the main lobes dimensions, that the study conducted with the higher analysis frequency gave a more precise result.

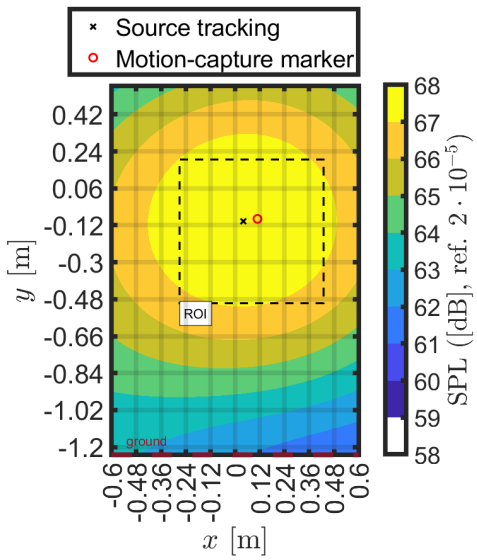

Figure 6.4: Analysis results of monopolar source experiment m1

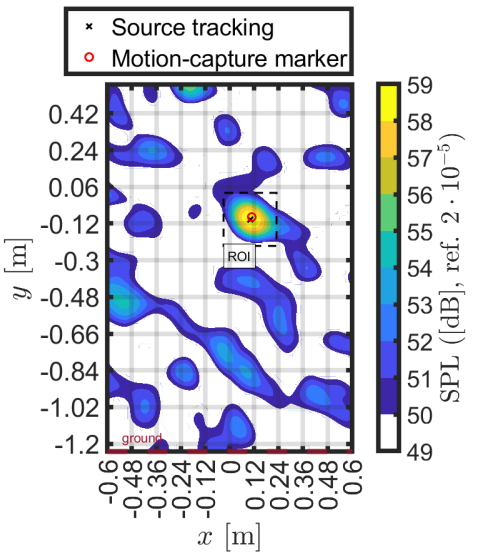

Figure 6.5: Analysis results of monopolar source experiment m2

About experiments m3 and m4, considering axial distances of $1m$ and $3m$ respectively, a constant analysis frequency of $3000Hz$ was chosen, by looking at the SPL levels in figure 6.3.

The results of the analysis performed on the third and fourth data-sets are showed in figures 6.6 and 6.7. Regarding this pair of recordings, it is noted that the experiment conducted by placing the source closer to the antenna shows greater precision in locating the source. As before, it is possible to examine the distance between the position with the maximum beamforming output and the optical marker, which is $14mm$ for the experiment n°3 and $39mm$ for the experiment n°4.

The results obtained are in agreement with what is expressed by the Rayleigh criteria regarding the analysis frequency and the distance between the source and

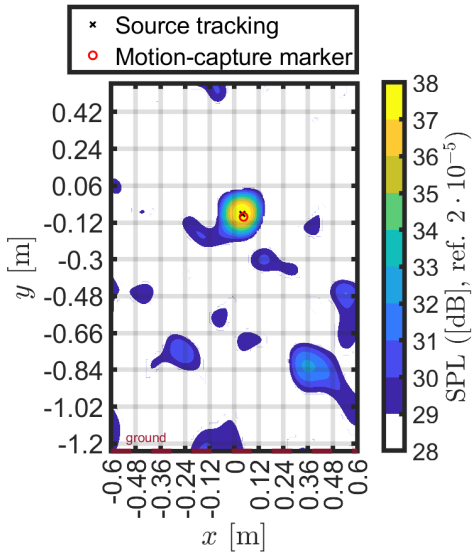

Figure 6.6: Analysis results of monopolar source experiment m3

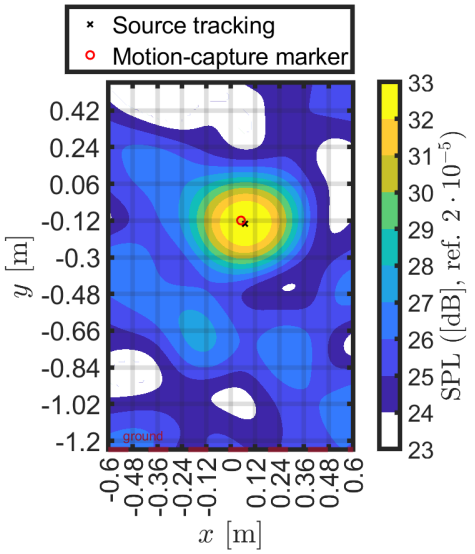

Figure 6.7: Analysis results of monopolar source experiment m4

the sensor array used [20]. Indeed, using a lower analysis frequency (associated to a longer wave length), it's more difficult to distinguish different nearby sound sources. On the other hand, if the distance between Scanning Grid and array rises, the potential sources are perceived by the system as closer to each other.

Both the choice of the analysis frequency and the distance between the Scanning Grid and the array can cause variations in the angular resolution of the system.

6.2 Tracking of hovering drones

In this section, two DTB analysis are performed on signal recorded about a hovering drone at the same height to the ground, and almost in the same position in front of the antenna. The two recordings differ in the orientation of the drone, which is showing to the array its front in the first data-set and its right side in the second data-set.

	Drone's orientation	Drone's average distance [m]	duration [s]
h1	Frontal	1.914	30
h2	Lateral	2.054	30

Table 6.2: Hovering drones test-matrix

6.2.1 Hovering drones: spectrograms and spectra

Spectrograms about these recordings are reported in figures 6.8 and 6.9, while in figures 6.10 and 6.11 particulars about a lower frequency range (between $100Hz$ and $1200HZ$) are showed.

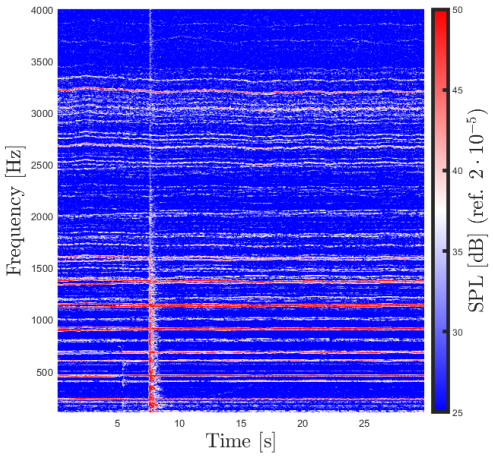

Figure 6.8: Spectrogram of experiment h1

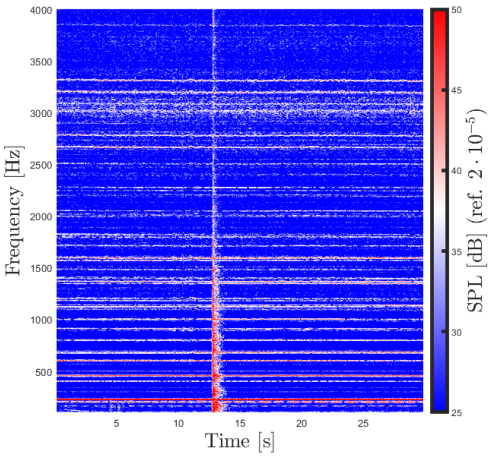

Figure 6.9: Spectrogram of experiment h2

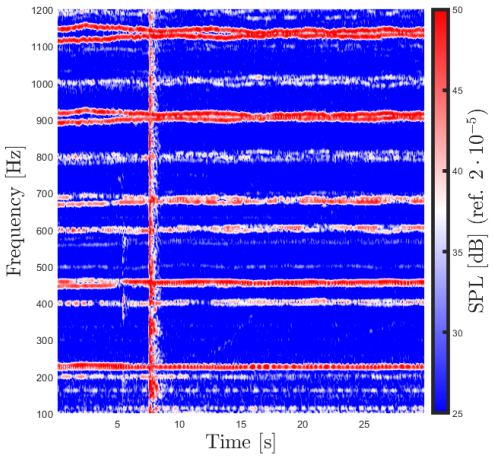

Figure 6.10: Particular of h1 experiment spectrogram

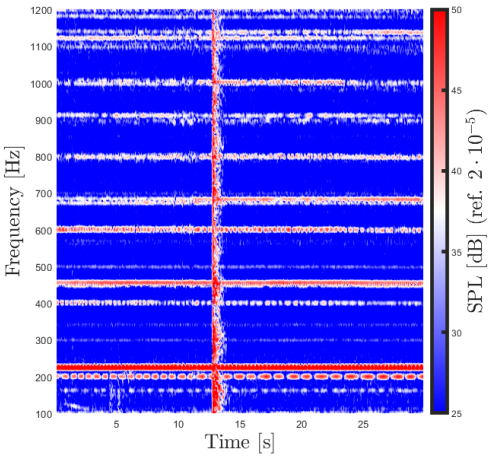

Figure 6.11: Particular of h2 experiment spectrogram

First of all, the spectrograms make clear the moment in which the sync-stick touches the ground, identified by a red vertical line, which includes almost all the frequencies on both graphs and affects especially the low frequencies. The BPFs are clearly visible and almost identical for both the experiments, around $200Hz$ and $225Hz$. There are two BPFs because the drone has two pairs of propellers which are mounted at different heights and which, in stable conditions, work with two different speeds, around $6000rpm$ and $6750rpm$.

By comparing the spectrogram of the two experiments, especially their zoom on the low frequencies, a note can be made about the shapes of the horizontal lines and about the stability of the flying drone. It is quite clear that lines in figure 6.11, about experiment h2, are more uniform than those in figure 6.10, about experiment h1. This is especially true for the harmonics around $1100Hz$, where even a line splitting can be noticed. This occurs when two rotors, normally having similar rotation speeds, stop balancing their own actions in order to exert a torque.

In the case of experiment h1, a drone's lateral balancing maneuver is probably observed. It is quite delicate as a maneuver, but shows that during the experiment h1 the drone was less stable than in the case of experiment h2, in which the horizontal lines of the spectrogram appear very uniform. This fact can be checked looking at the drone's average displacements, listed in table 6.3, which are really smaller between the chunks used in the DTB analysis of experiment h2.

Experiment	Chunks	$\Delta x[mm]$	$\Delta y[mm]$	$\Delta z[mm]$
h1	$1 \rightarrow 2$	0.85	-0.37	-0.61
	$2 \rightarrow 3$	1.15	-0.20	-0.39
	$3 \rightarrow 4$	1.24	-0.55	-0.28
h2	$1 \rightarrow 2$	-0.95	0.03	-0.19
	$2 \rightarrow 3$	-0.94	-0.06	0.05
	$3 \rightarrow 4$	-0.10	-0.09	-0.01

Table 6.3: Average drone displacements for experiments h1 and h2

This leads to the choice of proper sections of the time-series of the two experiments (in particular for experiment h1), where the spectrogram are uniform and not affected by the synchronization noise, so that the results of the DTB analysis can be more accurate. About experiment h1, a proper time interval is considered between $t = 16s$ and $t = 17s$; about experiment h2 time interval between $t = 25s$ and $t = 26s$ is preferred.

To decide with precision a common analysis frequency, two PSD spectra are computed for the considered sections of the time-series and then plotted (figures 6.12 and 6.13)

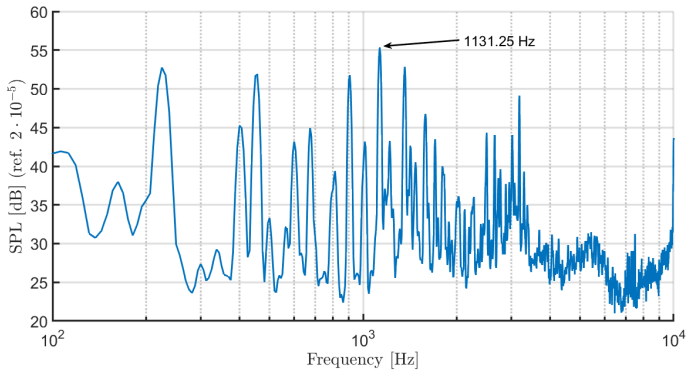

Figure 6.12: Spectrum of experiment h1: from $t = 16s$ to $t = 17s$

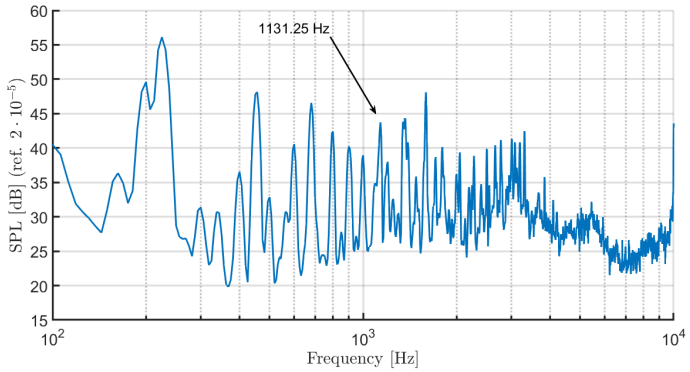

Figure 6.13: Spectrum of experiment h1: from $t = 25s$ to $t = 26s$

6.2.2 Hovering drones: source-maps

The following images show two sequences of beamforming outputs: figures 6.14, 6.15, 6.16 and 6.17 are about experiment h1, while figures 6.18, 6.19, 6.20 and 6.21 are about experiment h2.

Both the sequences show the ability of the DTB method in tracking a slowly moving distributed source as a drone in hover.

6.3 Tracking of a drone in motion

The last experiment presented is about a sequence of fast alternating lateral flights, with the drone oriented with its front to the array.

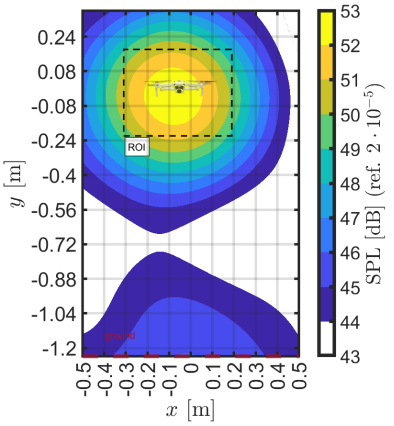

Figure 6.14: DTB on h1, chunk 1

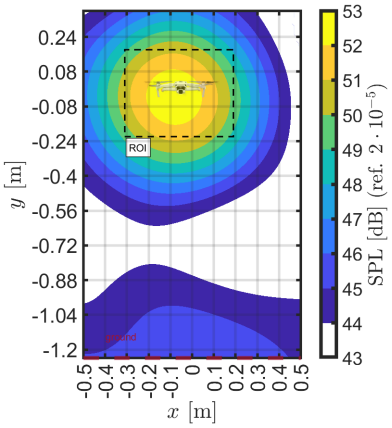

Figure 6.15: DTB on h1, chunk 2

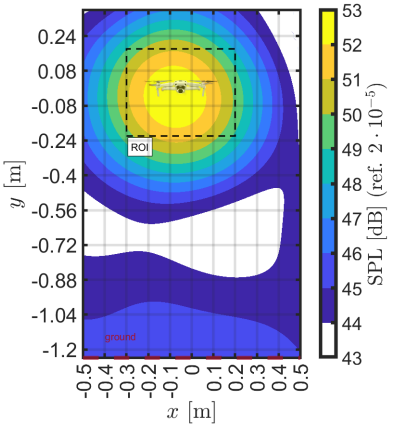

Figure 6.16: DTB on h1, chunk 3

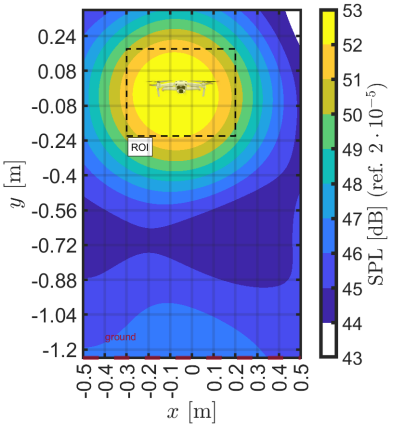

Figure 6.17: DTB on h1, chunk 4

6.3.1 Drone in motion: spectrograms and spectra

The spectrogram obtained by processing the data of the central microphone is plotted in figure 6.22 and shows an almost irregular behaviour of the BPFs and their harmonics, after the synchronization signal. However many vertical stripes can be observed, indicating temporary growths of the SPL, where the drone accelerates and noise emission rises, while covering a wider range of frequency domain because of the changes in propellers' rotation speed.

This is highlighted in figure 6.23, where a fragment of the previous spectrogram

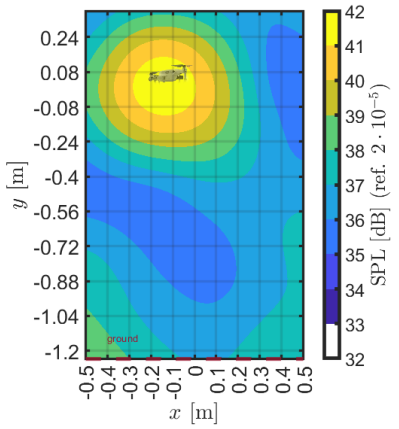

Figure 6.18: DTB on h2, chunk 1

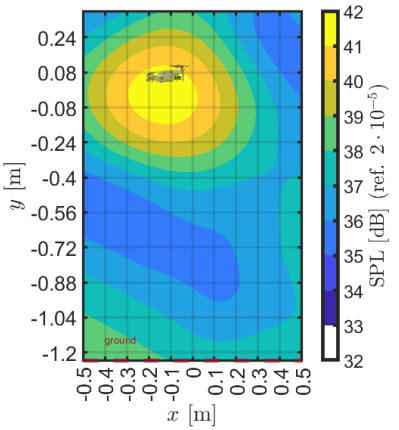

Figure 6.19: DTB on h2, chunk 2

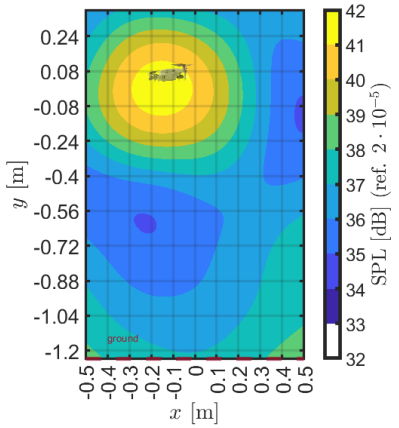

Figure 6.20: DTB on h2, chunk 3

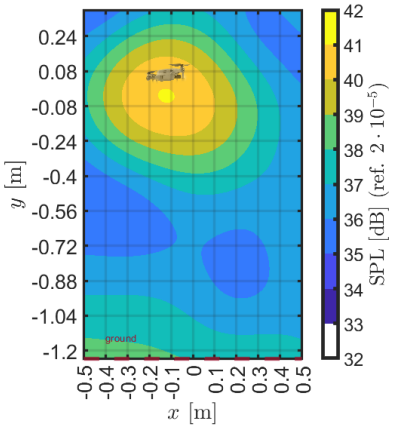

Figure 6.21: DTB on h2, chunk 4

is plotted and synchronized with the information about the x-component of the drone's trajectory. In the spectrogram fragment, only frequencies around the usual BPFs are considered and the SPL dynamic range is modified in order to make evident that every time the drone changes direction a fast variation in the most contributing frequencies takes place.

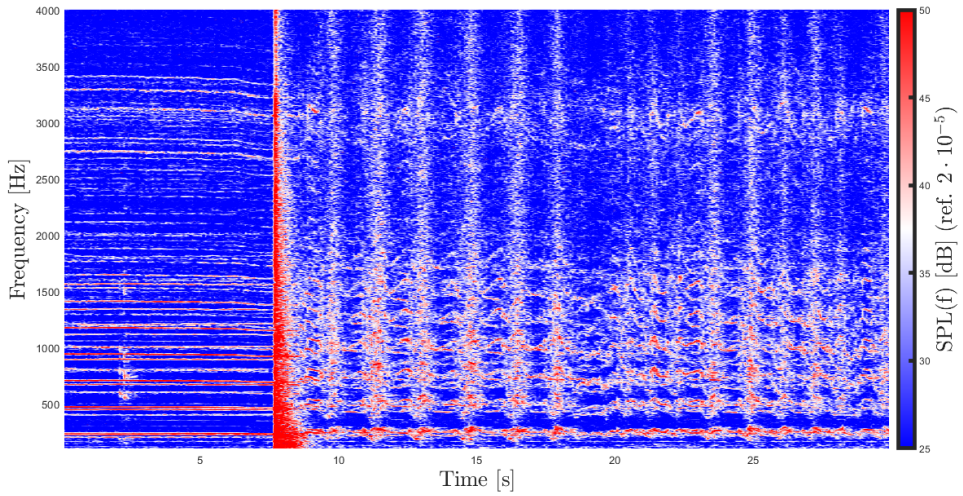

Figure 6.22: Spectrogram of a sequence of lateral maneuvers

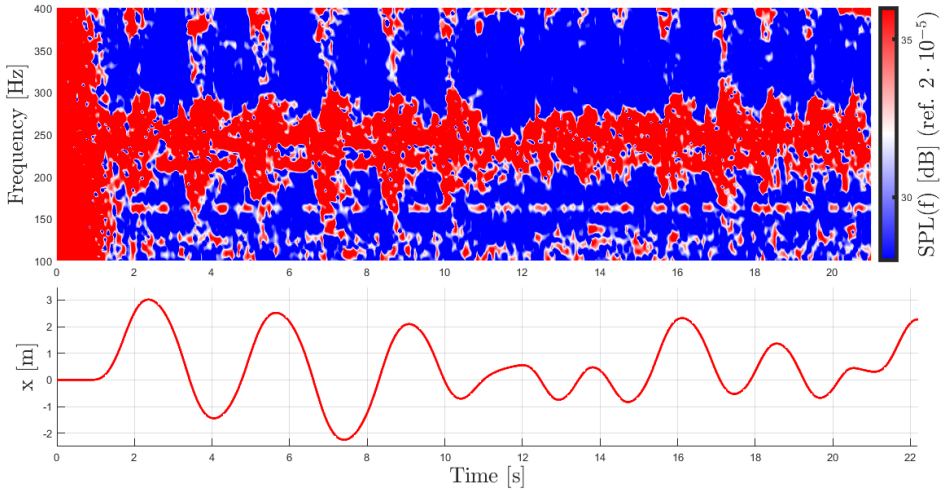

Figure 6.23: Spectrogram fragment synchronized with the x-component of drone's trajectory

6.3.2 Drone in motion: source maps

Two seconds of the whole recording previously showed are elected as sample to be analyzed with the DTB algorithm. The time interval selected goes from $t = 8s$ to $t = 10s$ after the sync-signal and includes a forward acceleration before a strong deceleration to invert drone's motion direction. The analysis is performed at a

frequency equal to 981.25Hz and the obtained results (showed in figures from 6.25 to 6.32) indicate a good quality of the drone tracking.

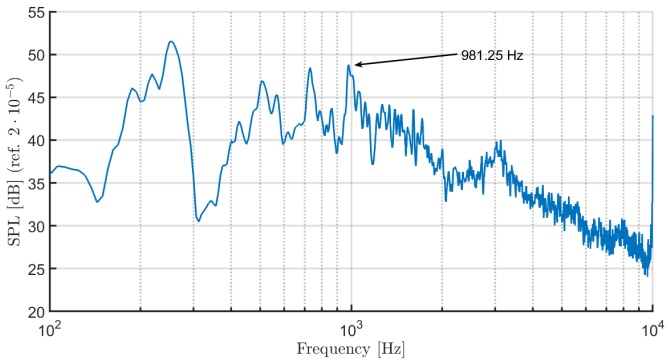

Figure 6.24: Spectrum

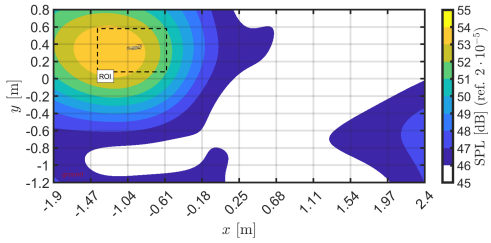

Figure 6.25: DTB on lf, chunk 1

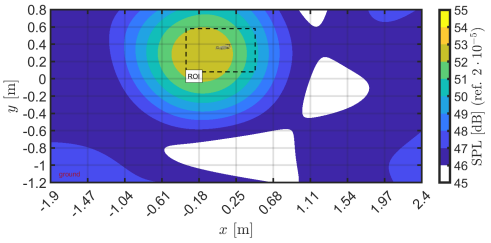

Figure 6.26: DTB on lf, chunk 2

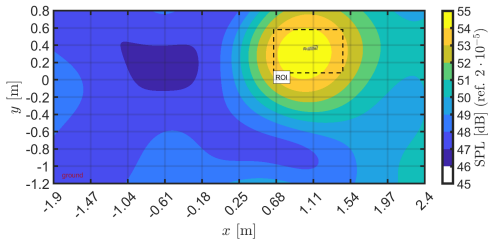

Figure 6.27: DTB on lf, chunk 3

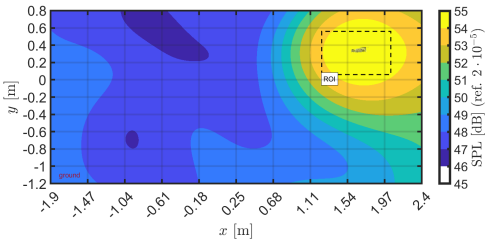

Figure 6.28: DTB on lf, chunk 4

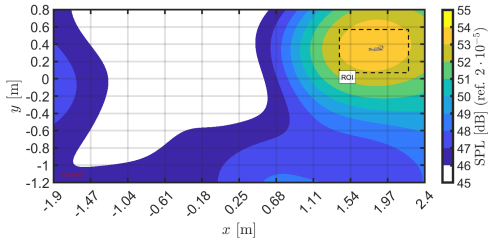

Figure 6.29: DTB on lf, chunk 5

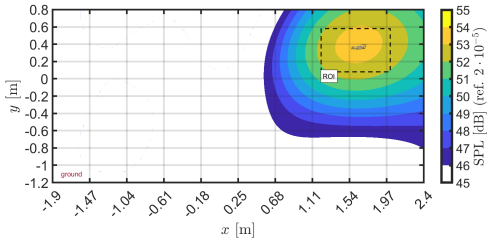

Figure 6.30: DTB on lf, chunk 6

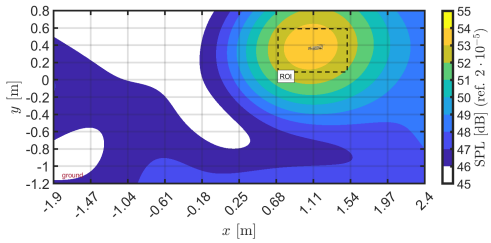

Figure 6.31: DTB on lf, chunk 7

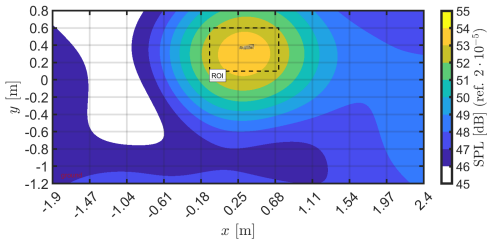

Figure 6.32: DTB on lf, chunk 8

6.3.3 Source Power Integration

Source Power integration is applied to the results of full spectrum beamforming analysis, in order to compare the acoustic field description, obtained with the DTB algorithm, with a classic spectrum, estimated on the recordings of the array central microphone.

Four experiments are considered for this analysis, for which the applied Regions Of Integration are showed in the corresponding figures of the previous sections.

SPI of monopolar sources

Results of SPI on experiments m1 and m2 are showed in figures 6.33 and 6.34.

For experiment m1, especially the fundamental frequency, the second and the third harmonics match their single-microphone counterparts.

Experiment m2 only allow the comparison between the fundamental and the second harmonics, being the third one localized beyond electrets' limit of accuracy, around $10kHz$. The first peak shows a good superimposition with the classic spectrum plot.

About the "base" levels of both the spectra comparisons, they do not show a complete overlapping. Somewhere, the SPI results are lower than those about single-microphone analysis: partially, this could be related to the fact that other sources composing the signal recorded by the single microphone could not be positioned in the ROI, not being considered by the SPI operations.

However, generically, the non-overlap of the spectra is not a negative aspect of these results: given that the method eliminates acoustic contributions due to environmental reverberation, this variation represents a positive result, as long as the difference, compared to the spectrum's levels, is not excessive.

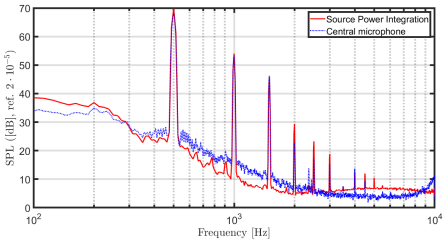

Figure 6.33: SPI on experiment m1

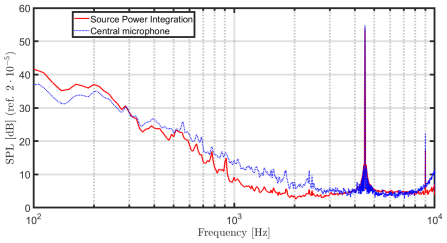

Figure 6.34: SPI on experiment m2

SPI on a hovering drone

For a drone in hover, the recording analyzed is the one about experiment h1, considering a drone oriented with its front facing the array. The signals were analyzed maintaining the same division in four chunks considered for the source-maps production.

The resulting SPI spectra, plotted and compared to the spectra of the respective signal chunks detected by the central microphones of the array, are showed in figures 6.35, 6.36, 6.37 and 6.38.

As expected, all the four plots are very similar to each other, being it a static flight. The reference single-microphone spectra are quite respected by the curves resulting from the SPI, especially at low frequencies. However as the analysis frequency rises the precision of overlapping between the spectra decreases.

SPI on a drone in motion

The last Source Power Integration analysis was performed on the chunks of the signal about the drone in lateral flight.

The results of this analysis (figures from 6.39 to 6.46) remark the difference between the acoustic emission of a hover and a maneuver with accelerations. Indeed,

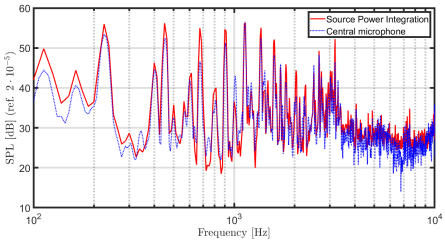

Figure 6.35: SPI on h1, chunk 1

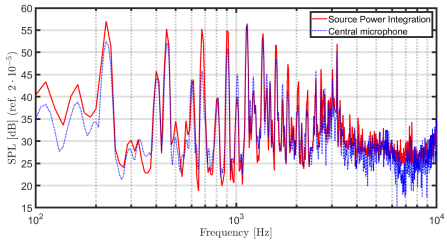

Figure 6.36: SPI on h1, chunk 2

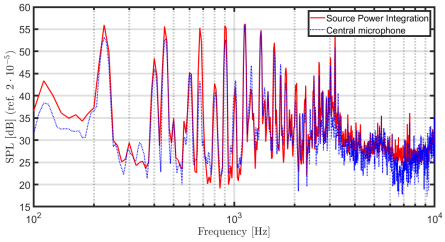

Figure 6.37: SPI on h1, chunk 3

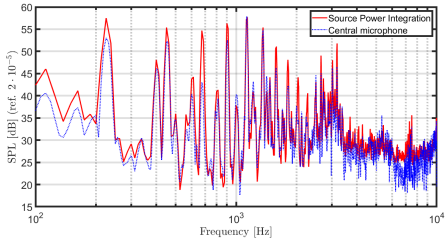

Figure 6.38: SPI on h1, chunk 4

in this case, it is difficult to make a comparison between the spectra about different chunks obtained with the SPI procedure, as it is with those obtained from the central microphone recordings.

In general, a clear feature of these graphs is represented by a reduced overlap of the curves, with respect to those about the hovering drone, highlighting the increased difficulty in studying a source with not-uniform motion.

Attention must be paid to this behaviour, whose origin could lie in the choice of the ROIs, rather than in the elimination of environmental reverberations. In fact, the center of the ROI doesn't coincides with the position of the source indicated by the DTB, but with the average position of the motion capture marker during the chunk. Since the drone could be very fast or accelerate suddenly in a certain instant of the chunk analyzed, the uniformity of the maps produced with the DTB method could be reduced, rising the difficulty of including in the ROIs many acoustic contributions that should not be overlooked.

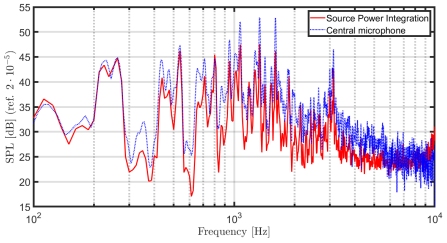

Figure 6.39: SPI on lf, chunk 1

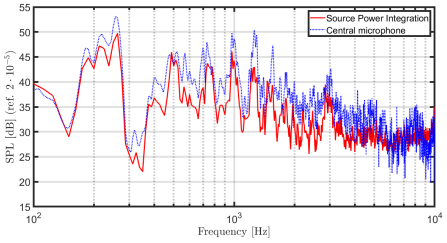

Figure 6.40: SPI on lf, chunk 2

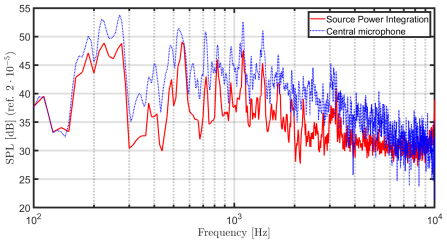

Figure 6.41: SPI on lf, chunk 3

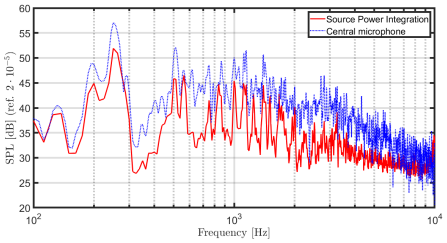

Figure 6.42: SPI on lf, chunk 4

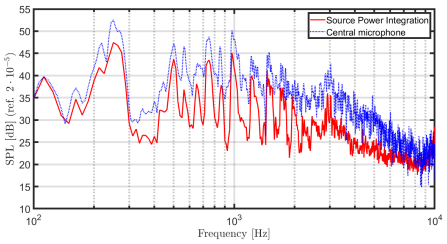

Figure 6.43: SPI on lf, chunk 5

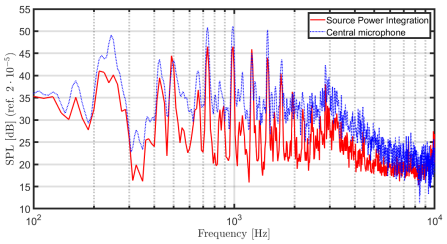

Figure 6.44: SPI on lf, chunk 6

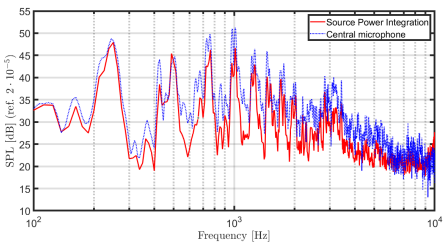

Figure 6.45: SPI on lf, chunk 7

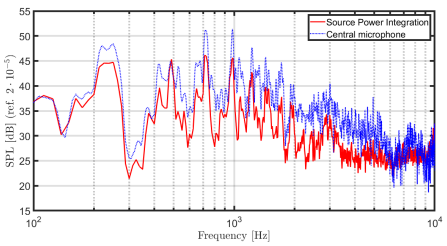

Figure 6.46: SPI on lf, chunk 8

Chapter 7

Conclusions and future perspectives

The work reported in this thesis was carried out at the von Karman Institute for Fluid Dynamics and shows the experience gained during the Short Training Programme, completed at its Environmental and Applied Fluid Dynamics Department.

With the aim at the validation of the proposed beamforming method, the activities undertaken were focused on the preparation of an experimental campaign, subsequently successfully completed, as well as on the processing of the gathered data.

The first period was completely dedicated to the development of a MATLAB code for a calibration procedure, whose output was applied on the signals of the electrets constituting the array, in order to enhance the quality of the data to be processed with the DTB method. The calibration procedure required the utilization of reference signals recorded at the VKI's facilities.

The two main experimental sessions took places in ID2MOVE test-hall, a wide room specifically designed for drones' flight, mounting a motion-capture system with a set of twelve cameras, essential for the validation of the tracking algorithm.

Concerning the post processing of the data, it was carried out using two software developed on MATLAB. The first code was used to create the source-maps, necessary to verify the tracking of a drone along its trajectory. The second code produced the spectra resulting from the Source Power Integration, to be compared with those about single microphones signals.

Results highlighted the capacity of the method in tracking a drone flying in front of the antenna at different speeds, as well as the possibility of defining a spectrum from the SPI, similar to anyone relative to a signal recorded by a single microphone of the array, but with the deletion of environmental reverberation's

spoiling contribution.

About the future perspectives, this method could be enhanced by refining the definition of the ROIs and applying psychoacoustic parameters to the results. Another step forward for this method would be the implementation and testing of the DTB using a double array. In this configuration, the second array would be positioned on a plane perpendicular to the first one, allowing a double source-map output to be combined in order to attain a 3D tracking of the sound sources. Other improvements would be the use of CLEAN-SC method on the current beamforming results or the implementation of the inverse beamforming method, instead of the conventional one, in the DTB algorithm. Both of them would enhance the accuracy of the maps as well as the spectra from the SPI method.

Appendix A

Electrets restoration

A.1 Functional restoration

To be sure that the microphones used for the experimental campaign would have worked properly, all of them were tested. Every step of the calibration have been inspected, to follow the transformations of the signals computed using the TFs.

Many electrets were broken or showed a signal impossible to superimpose on the reference one. Those were fixed or replaced by the VKI's electronic laboratory.

A.2 Structural restoration

Electrets consist of a small sensor directly linked to a connector by a wire so they lack structural integrity. As they were intended to be positioned in the Dougherty array, the microphones were inserted into brass tubes to give them stiffness. Then each pair of tubes and wires was secured together with heat-shrink tubes.

As this process had been previously used, during the microphones' check-up it was clear that plastic aging had led to the sliding of the old shrunk tube along the microphones' wire and, in some cases, along the brass tube.

To improve the grip between the wire and the new heat-shrink tube, a small portion of the old shrunk plastic was kept and included between the new plastic tube and the wire.

To prevent the old shrunk plastic to slide along the brass tube, for those microphones with a good grip between the old shrunk plastic tube and the wire, the whole piece of old plastic tube was kept, and some glue was applied on the brass tube to keep the old tube in position.

Appendix B

Calibration code

```
clear all %#ok<CLALL>
close all
clc

%% Folders and path

currentScript = mfilename('fullpath');
                        % .m file directory (string)
[folderstr,~,~] = fileparts(currentScript);
                        % .m folder directory (string)


calibrationDirectory = fullfile(folderstr,"..");
                        % Main directory for the
   calibration (string)
dataFolder = fullfile(calibrationDirectory + '\Dati
   calibrazione Nivelles 2023_07_12');
       % Folder with data for calibration

% Path
addpath(calibrationDirectory + '\TDMS_package');
                        % Add TDMS package to the current
    path

%% Font definition

```

```
19 fontname = 'Times';
20 fontsize = 16;
21 set(0,'defaultaxesfontname',fontname);
22 set(0,'defaultaxesfontsize',fontsize);
23 set(0,'defaulttextfontname',fontname);
24
25 clear fontname fontsize
26
27 %% Acquisition parameters, reference values and
      dataprocessing parameters
28
29 fs = 51200;
                          % Sampling frequency
30
31 ts = 30;
                          % Acquisition time
32 data_length = fs*ts;
33
34 N_mic = 64;
                          % Microphone's number
35
36 filterBand = [1e2 1e4];
                          % Frequency range for bandpass
      filter application
37
38 % Parameters for the frequency domain analysis
39 window = 65536;
40 noverlap = window/2;
41 nfft = window*2;
42
43 %% Definizione matrici contenenti i dati e correzione
      con sensitivity dei dati da B&K
44
45 load B&Ksensitivity
                          %[v/Pa], calcolata con
      pistonphone
46
47 % Name root -----> specify the bottom plate mic --->
      elec: electret; flush: flush mounted B&K)
48 % Name suffix ---> specify data origin ------------> (
      Ele: electret; Ref: reference flush mounted B&K;
```

```
49 %
       CalLow: B&K in the lower calibrator side hole;
50 %
       CalUp: B&K in the upper calibrator side hole.
51
52 fileName = fullfile(dataFolder,"\B&K_Ref.tdms");
53 fileName = convertStringsToChars(fileName);
                          % Convert string to char as
       TDMS_readTDMSFile doesn't read strings (?)
54
55 fileStruct = TDMS_readTDMSFile(fileName);
56
57 flushRef = fileStruct.data{1,4}/BnKSensitivity;
58 flushRef = bandpass(flushRef,filterBand,fs);
59 flushCalLow = fileStruct.data{1,5}/BnKSensitivity;
60 flushCalLow = bandpass(flushCalLow,filterBand,fs);
61 flushCalUp = fileStruct.data{1,6}/BnKSensitivity;
62 flushCalUp = bandpass(flushCalUp,filterBand,fs);
63
64 % Preallocation
65 elecEle = zeros(N_mic,data_length);
66 elecCalLow = zeros(N_mic,data_length);
67 elecCalUp = zeros(N_mic,data_length);
68
69 for i = 1:N_mic
70
71     fprintf(string(i + "\n"));
72
73     if (i<10)
74
75         fileName = fullfile(dataFolder,"Ele_0"+ i +".
       tdms");
76         fileName = convertStringsToChars(fileName);
77
78         % Collecting data from the tdms files ->
       creating 3 matrices
79         fileStruct = TDMS_readTDMSFile(fileName);
80
81         % A row for each microphone
82         elecEle(i,:) = fileStruct.data{1,4};
```

```
83          elecEle(i,:) = bandpass(elecEle(i,:),filterBand,
    fs);
84
85          elecCalLow(i,:) = fileStruct.data{1,5};
86          elecCalLow(i,:) = bandpass(elecCalLow(i,:),
    filterBand,fs);
87
88          elecCalUp(i,:) = fileStruct.data{1,6};
89          elecCalUp(i,:) = bandpass(elecCalUp(i,:),
    filterBand,fs);
90
91      else
92
93          fileName = fullfile(dataFolder,"Ele_"+ i +".tdms
    ");
94          fileName = convertStringsToChars(fileName);
95
96          % Collecting data from the tdms files ->
    creating 3 matrices
97          fileStruct = TDMS_readTDMSFile(fileName);
98
99          elecEle(i,:) = fileStruct.data{1,4};
100         elecEle(i,:) = bandpass(elecEle(i,:),filterBand,
    fs);
101
102         elecCalLow(i,:) = fileStruct.data{1,5};
103         elecCalLow(i,:) = bandpass(elecCalLow(i,:),
    filterBand,fs);
104
105         elecCalUp(i,:) = fileStruct.data{1,6};
106         elecCalUp(i,:) = bandpass(elecCalUp(i,:),
    filterBand,fs);
107
108     end
109
110 end
111
112
113 elecCalLow = elecCalLow/BnKSensitivity;
114 elecCalUp = elecCalUp/BnKSensitivity;
115
```

```
116 clear currentFile cal_dir fileName folderstr
       calibrationDirectory
117 save Signals flushRef elecEle flushCalLow flushCalUp
       elecCalLow elecCalUp
118
119 %% Transfer functions' computation
120
121 [tf_RefCalLow,f_tf] = tfestimate(flushRef,flushCalLow,
       window,noverlap,nfft,fs,"onesided");
122
123 tf_RefCalUp = tfestimate(flushRef,flushCalUp,window,
       noverlap,nfft,fs,"onesided");
124
125 % Preallocation
126 tf_CalLowEle = zeros(length(tf_RefCalLow),N_mic);
127 tf_CalUpEle = zeros(length(tf_RefCalLow),N_mic);
128
129 for i = 1:N_mic
130
131     tf_CalLowEle(:,i) = tfestimate(elecCalLow(i,:),
       elecEle(i,:),window,noverlap,nfft,fs,"onesided");
132
133     tf_CalUpEle(:,i) = tfestimate(elecCalUp(i,:),elecEle
       (i,:),window,noverlap,nfft,fs,"onesided");
134
135 end
136
137 % Expansion of Ref vectors to get the right size
138 tf_RefCalLow = repmat(tf_RefCalLow,1,N_mic);
139 tf_RefCalUp = repmat(tf_RefCalUp,1,N_mic);
140
141 tf_final_CalLow = tf_CalLowEle.*tf_RefCalLow;
142 tf_final_CalUp = tf_CalUpEle.*tf_RefCalUp;
143
144 %% Composition of the 2 final TF to cut out resonances'
       effects
145
146 % Preallocation
147 TFM = zeros(size(tf_final_CalLow,1),N_mic);
148
149 for i = 1:N_mic
```

```
150 tf_mergedVec = TFM(:,i);
151 tf_final_CalLowVec = tf_final_CalLow(:,i);
152 tf_final_CalUpVec = tf_final_CalUp(:,i);
153
154 tf_mergedVec(f_tf<80) = tf_final_CalUpVec(f_tf<80);
155 tf_mergedVec(f_tf>=80) = tf_final_CalLowVec(f_tf>=80);
156
157 tf_mergedVec(f_tf<4000) = tf_mergedVec(f_tf<4000);
158 tf_mergedVec(f_tf>=4000) = tf_final_CalUpVec(f_tf>=4000)
    ;
159
160 tf_mergedVec(f_tf<8000) = tf_mergedVec(f_tf<8000);
161 tf_mergedVec(f_tf>=8000) = tf_final_CalLowVec(f_tf
    >=8000);
162 TFM(:,i) = tf_mergedVec;
163 end
164
165 %% Matrices' saving
166
167 tf_RefCalLow(:,2:end) = [];
168 tf_RefCalUp(:,2:end) = [];
169
170 save TF_matrices.mat tf_RefCalLow tf_RefCalUp
    tf_CalLowEle tf_CalUpEle tf_final_CalLow
    tf_final_CalUp N_mic
171 save TFM.mat TFM f_tf
172 %% Plots
173
174 for i = 1:N_mic
175
176     figure(i)
177
178     subplot(2,1,1)
179     semilogx(f_tf,abs(tf_final_CalLow(:,i)),'r','
    linewidth',2)
180     hold on
181     semilogx(f_tf,abs(tf_final_CalUp(:,i)),'b','
    linewidth',2)
182     hold off
183
184     grid on
```

```
185     grid minor
186     xlim([1e1 1e4])
187 %     ylim([0 1])
188     xticklabels({})
189     ylabel("|TF|","Interpreter","latex")
190 
191     legend("B&K_{R} \rightarrow B&K_{CalLow} \rightarrow El", ...
192     "B&K_{R} \rightarrow B&K_{CalUp} \rightarrow El","Location","northwest")
193 
194     title("Microphone n^o " + i)
195 
196 
197 
198     subplot(2,1,2)
199 %     subplot(2,1,2)
200     semilogx(f_tf,unwrap(atan2(imag(tf_final_CalLow(:,i)),real(tf_final_CalLow(:,i)))),'r','linewidth',2)
201     hold on
202     semilogx(f_tf,unwrap(atan2(imag(tf_final_CalUp(:,i)),real(tf_final_CalUp(:,i)))),'b','linewidth',2)
203     hold off
204     grid on
205     xlim([1e1 1e4])
206 %     xticklabels({})
207     yticks(-5*pi:pi:5*pi)
208     ii=1;
209     for j = -5:5
210         if i==0
211             ichar(ii) = num2str(j); %#ok<SAGROW>
212         else
213             ichar(ii) = num2str(j) + "\pi"; %#ok<SAGROW>
214         end
215     ii=ii+1;
216     end
217     yticklabels(ichar)
218     ylabel("$\angle_{TF} [rad]$","Interpreter","latex")
219     xlabel("Frequency [Hz]","Interpreter","latex")
220 end
```

Appendix C

DTB code

```
clear all %#ok<CLALL>
close all
clc

load filesAndFolders.mat
load signal.mat
load camData.mat
load info

dataFileName = extractBefore(dataFileName,".tdms");
dataFileNameForPlotTitle = strrep(dataFileName,"_","\_")
    ;

%% Analysis configuration

analysisFrequency = 1000;        % [Hz]; analysis
    frequency
chunkNumber = 8;                 % Number of time
    segments in which the signal is divided

droneFollowing = 2;
droneType = 2;  % 1 -> Phanom 3;    2 -> Air 2S
droneOrientation = 1;   % 1 -> front;   2 -> side (front
     to the right)
xlength = 1.6; % [m] Horizontal dimension of SG
ylength = 1.6; % [m] Vertical dimension of SG
```

```
23 xSpacing = 0.01; % [m] spacing for horizontal SG
       dimension
24 ySpacing = 0.01; % [m] spacing for vertical SG dimension
25
26 automaticMapDynamicRange = 1;
27 mapMaxVal = 42;%50
28 mapMinVal = 32;%40
29 mapStepNum = 9;
30 mapTickGap = 1;
31
32 % Analysis on third octave wide frequency bands
33 third_oct_band = 1;                  % 0 => TOB analysis
       is off;     1 => TOB analysis is on
34
35 % CSM diagonal removal analysis
36 diagonal_removal = 0;                % 0 => DR analysis
       is off;      1 => DR analysis is on
37
38 % Functional Beamforming analysis
39 nu = 1;                              % nu = 1 => CB
       method is applied to time chunks;    nu>1 => FB
       method is applied to time chunks
40
41 SteeringVectorFlag = 0;           % 0 => formulation g/
       norm(g)^2;    1 => formulation g/norm(g)
42
43
44 % Analysis with reflections
45 %-------------------------------------------
46 % Mirror
47 %   A: activation;  [B]: direction normal to the mirror;
       [C]: mirror's position
48 %   mirror = [A,      [B],         [C]];
49     mirror = [0, [0,-1,0], [0,-1.25,3]];    % A=0 =>
       mirror off;    A=1 => mirror on
50
51 % Duct
52 %   A: activation;  B: activation of image method;  [C]:
       duct's dimensions
53 %   Duct = [A, B,   [C]];                   [C] = [x:
       Depth, y: Height]     =>  z is infinite
```

```
54      Duct = [0, 1, [8,8]];                          % A=0 =>
    duct off;       A=1 => duct on
55
56  % Room
57  %   A: activation;  B: activation of image method;  [C]:
     Rooms's dimensions
58  %   Room = [A, B,   [C]]                           [C] = [x:
    Depth, y: Height, z:Length]
59  Room = [0,1,[23.6,8.5,26.8]];                      % A=0 =>
    room off;       A=1 => room on
60                                                     % B=0 =>
    modal method;  B=1 => image method
61
62  order = 2;  % Number of reflection considered by the
    analysis
63  Coeff = 1;  % Absorption coefficient of the reflective
    wall (?)
64  %
    ------------------------------------------------------------
65
66  Source_type = 1;
67  d_dipole = [0, 0, 1];
68
69  %% MATLAB Path
70
71  addpath(filesAndFolders.scanningGridFolder)
72
73  %% Windowing parameters
74
75  window = 4096;
76  noverlap = 0.5*window;
77  nfft = window*2;
78  fs = 51200;
79
80  %% Ambient conditions
81
82  temp = 273.15 + 20;  % [K]; Atmosphere temperature
83  c = 331.5 + (temp-273.15)*(3/5);  % [m/s]; speed of
    sound
84
```

```
85 %% Preallocation of main variables
86
87 SPL_overall = zeros(chunkNumber,1);
88 SPL_DTB = zeros(chunkNumber,1);
89 SPL_analysisFrequency = zeros(chunkNumber,length(
      analysisFrequency));
90
91 dronePositionMemory = zeros(chunkNumber,3);
92
93 x_pos= zeros(1,chunkNumber);
94 y_pos = zeros(1,chunkNumber);
95
96 FN = fieldnames(camData.dronePoints);
97
98 tic
99
100 %% Array
101
102 Array = readmatrix(filesAndFolders.arrayFile);
103 N_mic = size(Array,1);
104
105 %%
106 for currentChunk = 1:chunkNumber
107
108     %% Position and velocity of the drone from Motion-
      capture system
109
110     xDroneChunk = trajectoryChunk(camData.dronePoints.(
      FN{1})(:,1), chunkNumber, currentChunk);
111     nFramePerChunk = length(xDroneChunk);
112     timeForVelComputation = (nFramePerChunk-1)*0.01;
113     xVelDroneChunk = (xDroneChunk(end)-xDroneChunk(1))/
      timeForVelComputation;
114     xDroneChunk = mean(xDroneChunk);
115
116     yDroneChunk = trajectoryChunk(camData.dronePoints.(
      FN{1})(:,2), chunkNumber, currentChunk);
117     yVelDroneChunk = (yDroneChunk(end)-yDroneChunk(1))/
      timeForVelComputation;
118     yDroneChunk = mean(yDroneChunk);
119
```

```
120     % Distance [m], along z axis, of the Scanning Grid
        from the origin (centre of the Dougherty array)
121     zDroneChunk = trajectoryChunk(camData.dronePoints.(
        FN{1})(:,3), chunkNumber, currentChunk);
122     zVelDroneChunk = (zDroneChunk(end)-zDroneChunk(1))/
        timeForVelComputation;
123     zDroneChunk = mean(zDroneChunk);
124
125     dronePosition_chunk_camera = [xDroneChunk
        yDroneChunk zDroneChunk];
126
127     dronePositionMemory (currentChunk,:) =
        dronePosition_chunk_camera;
128
129     if droneType==2
130         if droneOrientation == 1
131             Gdist = zDroneChunk-(0.180/2);%zDroneChunk
        refers to the marker positioned on top of drone...
        this way the scanning grid is not centered on the
        marker but on the nearest couple of propellers
132         elseif droneOrientation == 2
133             Gdist = zDroneChunk-(0.253/2+0.03);%
        zDroneChunk refers to the marker positioned on top of
         drone... this way the scanning grid is not centered
        on the marker but on the nearest couple of propellers
134         end
135     else
136         Gdist = zDroneChunk;
137     end
138
139     %% Chunk time
140
141     tBeginChunck = (currentChunk-1)*nFramePerChunk*0.01+
        timeVecVideo(1);
142     tEndChunck = tBeginChunck-0.01+nFramePerChunk*0.01;
143
144     %% Velocity
145
146     MachVector = [xVelDroneChunk/c, yVelDroneChunk/c,
        zVelDroneChunk/c];      % [-]
147
```

```
148     Beta = sqrt(1-norm(MachVector)^2);  % [-]
149
150     %% Scanning Grid
151
152     if droneFollowing == 1
153         [SGX,SGY,SGZ,x,y] = SG_xyzMoving(xDroneChunk,
    yDroneChunk,Gdist,xlength,ylength,xSpacing,ySpacing);
154     elseif droneFollowing == 0
155         [SGX,SGY,SGZ,x,y] = SG_zMoving(Gdist, xlength,
    ylength,xSpacing,ySpacing,groundHeight);
156 %         [SGX,SGY,SGZ,x,y] = SG_zMoving(Gdist, xlength,
     ylength,xSpacing,ySpacing,groundHeight,droneType,
    droneOrientation,camData);
157     elseif droneFollowing == 2
158         [SGX,SGY,SGZ,x,y,xlength,ylength,xExtremities,
    yExtremities] = SG_zMovingAutomaticWidth(Gdist,
    xSpacing,ySpacing,groundHeight,camData);
159     end
160
161     xnode = length(x);
162     ynode = length(y);
163     N_node = xnode*ynode;
164
165     for currentFrequency=1:length(analysisFrequency)
166         %% Analysis
167         % Cross Spectral Matrix computation
168         [CSM, F, Signal_chunk] = CSM_computation(
    acousticSignals, chunkNumber, currentChunk, window,
    noverlap, nfft, fs);
169         % The result of CSM_computation is already
    multiplied with the frequency bin
170
171         if third_oct_band == 1
172             % Alternative 1) Preparation of the one-
    third octave band analysis
173             fcentre = 10^3*(2.^((-18:13)/3));
174             fd = 2^(1/6);
175             fupper = fcentre*fd;
176             flower = fcentre/fd;
177             err = 10^3;
178             for i = 1:length(fcentre)
```

```
179                 delta = abs(fcentre(i)-analysisFrequency
    (currentFrequency));
180                 if delta < err
181                     err = delta;
182                     index = i;
183                 end
184             end
185             f_up = interp1(F,F,fupper,'nearest');
186             f_low = interp1(F,F,flower,'nearest');
187             f_pos_low = find(F==f_low(index));
188             f_pos_up = find(F==f_up(index));
189         else
190             % Alternative 2) Preparation of the narrow
    band analysis
191             f = interp1(F,F,analysisFrequency(
    currentFrequency),'nearest');
192             f_pos_low = find(F==f);
193             f_pos_up = find(F==f);
194         end
195
196
197         CBoutput_tot = zeros(ynode,xnode,size(f_pos_up-
    f_pos_low,2));
198         cnt = 1;
199
200         for f_pos = f_pos_low:f_pos_up
201             omega = 2*pi*F(f_pos);
202             CSMm = squeeze(CSM(:,:,f_pos));
203             if diagonal_removal == 1
204                 CSMm = CSMm-diag(diag(CSMm));
205             end
206
207             % Preallocating the CB's output
208             CBoutput = zeros(ynode,xnode);
209
210             % Green's Function and Steering Vector
    computation
211             disp('- Steering vector computation');
212             for xx = 1 : xnode
213                 for yy = 1 : ynode
214                     xn = SGX(yy, xx);
```

```
215                     yn = SGY(yy, xx);
216                     zn = SGZ(yy, xx);
217
218                     [PG] = SteeringVector(xn, yn, zn,
    Array, mirror, Duct, Room, MachVector, Beta, omega, c
    , N_mic, order, Coeff, Source_type, d_dipole);
219
220                     if SteeringVectorFlag == 0
221                         w = PG/norm(PG)^2;
222                     else
223                         w = PG/(norm(PG));%*sqrt(N_mic))
    ;
224                     end
225
226
227                     % CB's output computation
228                     CBoutput(yy,xx) = (w'*(CSMm^(1/nu))*
    w)^nu;
229                 end
230             end
231
232             CBoutput_tot(:,:,cnt) = CBoutput;
233
234             if third_oct_band == 1
235                 disp(['- Computation of f = ',num2str(F(
    f_pos),5),' Hz, ',num2str(100*(cnt-1)/(f_pos_up-
    f_pos_low),4),' % of the frequency band processed']);
236             end
237
238             cnt = cnt+1;
239             clear CBoutput;
240
241         end
242
243         %% Processing of CB's output
244         CBoutput = abs(sum(CBoutput_tot,3));
245
246         distRef = 1;
247
248         maximum = 0;
249         for xx = 1:xnode
```

```
250             for yy = 1:ynode
251                 if CBoutput(yy, xx) > maximum
252                     maximum = CBoutput(yy, xx);
253                     xxmax = xx;
254                     yymax = yy;
255                 end
256                 if third_oct_band==1
257                     CBoutput(yy, xx) = 10*log10(
    CBoutput(yy, xx)*F(2)/(4*pi*distRef*2e-5)^2)-10*log10
    (f_pos_up-f_pos_low);
258                 else
259                     CBoutput(yy, xx) = 10*log10(
    CBoutput(yy, xx)*F(2)/(4*pi*distRef*2e-5)^2);
260                 end
261             end
262         end
263         if third_oct_band==1
264             maximum = 10 * log10(maximum*F(2)/(4*pi*
    distRef*2e-5)^2)-10*log10(f_pos_up-f_pos_low);
265         else
266             maximum = 10 * log10(maximum*F(2)/(4*pi*
    distRef*2e-5)^2);
267         end
268
269         % Maximum SPL definition and collocation
270         SPL_DTB(currentChunk) = maximum;
271         x_pos(currentChunk) = xxmax;
272         y_pos(currentChunk) = yymax;
273
274         % Setting of the source-map's minimum threshold
275         for xx = 1 : xnode
276             for yy = 1 : ynode
277                 if CBoutput(yy,xx) < maximum-10
278                     CBoutput(yy,xx) = 0;
279                 end
280             end
281         end
282
283         %% Building of the source-map
284         X = SGX;
285         Y = SGY;
```

```
286         Z = CBoutput;
287 
288         if currentChunk < 10
289             savingIndex = char("_chunk_0" + currentChunk
    );
290         else
291             savingIndex = char("_chunk_" + currentChunk)
    ;
292         end
293 
294         NameSM = [filesAndFolders.mainFolder '\Results\
    SourceMaps_mat\frequency_' char(string(
    analysisFrequency(currentFrequency))) savingIndex '.
    mat'];
295         save(NameSM,'X', 'Y', 'Z')
296         writematrix( Z, [filesAndFolders.mainFolder '\
    Results\SourceMaps_txt\frequency_' char(string(
    analysisFrequency(currentFrequency))) savingIndex '.
    txt'])
297 
298         %% Plot source-map
299 
300         figure('units','normalized','outerposition',[0
    0.1 0.75 0.90])
301         hold on
302         pcolor(X, Y, Z)
303         shading('interp')
304         cm = [1 1 1; parula(mapStepNum)];
305         colormap(cm);
306 
307         % Colorbar
308         hcb = colorbar;
309         if automaticMapDynamicRange == 1
310             mapMinVal = round(max( max(Z))) - 10;
311             mapMaxVal = round( max( max(Z)));
312         end
313         clim([mapMinVal, mapMaxVal])
314         ylabel( hcb,'SPL ([dB], ref. $2 \cdot 10^{-5}$)'
    ,'FontSize',20,'Interpreter','latex')
315         set(hcb,'XTick',mapMinVal:mapTickGap:mapMaxVal,'
    linewidth',3.5)
```

```
316
317         % Axis
318         xlabel('$x$ [m]','Interpreter','latex')
319         ylabel('$y$ [m]','Interpreter','latex')
320         if mirror(1,1) == 1
321             line([-1 1],[mirror(6) mirror(6)],'Color','
    white','LineStyle','--','LineWidth',2.5)
322         end
323         axis equal
324         if droneFollowing == 0
325             xlim([-xlength/2 xlength/2])
326             xticks(-xlength/2:xlength/10:xlength/2);
327             ylim([- groundHeight, -groundHeight+ylength
    ])
328             yticks(round(-groundHeight,1,TieBreaker="
    minusinf"):ylength/10: round(-groundHeight+ylength,1,
    TieBreaker="minusinf"))
329         elseif droneFollowing == 1
330             xlim([-xlength/2 xlength/2]+xDroneChunk)
331             xticks(round(-xlength/2+xDroneChunk-0.1,1,
    TieBreaker="minusinf"):xlength/10:round(xlength/2+
    xDroneChunk-0.1,1,TieBreaker="minusinf"))
332             ylim([-ylength/2, ylength/2]+yDroneChunk
    -0.1)
333             yticks(round(-ylength/2+yDroneChunk-0.1,1,
    TieBreaker="minusinf"):ylength/10:round(ylength/2+
    yDroneChunk-0.1,1,TieBreaker="minusinf"))
334         elseif droneFollowing == 2
335             xlim(xExtremities)
336             xticks(xExtremities(1):xlength/10:
    xExtremities(2))
337             ylim(yExtremities)
338             yticks(yExtremities(1):ylength/10:
    yExtremities(2))
339         end
340         ax = gca;
341         ax.FontSize = 20;
342         box on
343         set(gca,'linewidth',2.5)
344         set(gca,'layer','top')
345         grid on
```

```
346
347         % Ground line
348         yline(-groundHeight,'--k','LineWidth',4,"Color
    ",[0.6350 0.0780 0.1840]);
349         if droneFollowing == 2
350             text(xExtremities(1)+0.1,yExtremities(1)
    +0.1,"ground","Color",[0.6350 0.0780 0.1840])
351         else
352             text(-xlength/2+0.1,-groundHeight+0.1,"
    ground","Color",[0.6350 0.0780 0.1840])
353         end
354
355         % Overlaying the drone image on the source-map
356         if droneType==2 && droneOrientation==1
357             [img, ~, tr] = imread('Air2sFront.png');
358             im = image('CData',img,'XData',[xDroneChunk
    -0.20 xDroneChunk+0.155],'YData',[yDroneChunk
    yDroneChunk-0.087]);
359             im.AlphaData = 0.7*tr;
360         elseif droneType==2 && droneOrientation==2
361             [img, ~, tr] = imread('Air2sSide.png');
362             im = image('CData',img,'XData',[xDroneChunk
    -0.101 xDroneChunk+0.119],'YData',[yDroneChunk+0.01
    yDroneChunk-0.07]);
363             im.AlphaData = 0.7*tr;
364         end
365
366         % Saving outputs
367         analysisFrequencyString = num2str(
    analysisFrequency(currentFrequency));
368         analysisFrequencyString = strrep(
    analysisFrequencyString,".","_");
369
370         saveas(gcf,[filesAndFolders.mainFolder '\Results
    \png\' char(dataFileName) '_analysisFreq_' char(
    analysisFrequencyString) char(num2str(savingIndex))],
    'png')
371         saveas(gcf,[filesAndFolders.mainFolder '\Results
    \fig\' char(dataFileName) '_analysisFreq_' char(
    analysisFrequencyString) char(num2str(savingIndex))],
    'fig')
```

```
372
373         % Control parameters
374         SPL_overall(currentChunk) = 20*log10(rms(
    Signal_chunk(end,:))/2e-5);       % [dB] array's
    central electret's overall SPL computation
375
376         [pxx,FRE] = pwelch(Signal_chunk(end,:),window,
    noverlap,nfft,fs);
377         SPL_analysisFrequency(currentChunk,
    currentFrequency) = 10*log10((pxx(FRE==
    analysisFrequency(currentFrequency))*(FRE(2)))/(2e-5)
    ^2); % Questo sembra funzionare. Nota che, essendo
    FRE(1)=0, Fre(2) corrisponde al Delta(FRE)=FRE(25)-
    FRE(24)
378
379       end
380 end
381
382 drone_displacement = dronePositionMemory(2:end,:)-
    dronePositionMemory(1:end-1,:);
383
384 cb_calc_time = toc;
385 disp(['Beamforming calculated on ',num2str(xnode*ynode),
    ' points in ',num2str(floor(cb_calc_time/60)),' min
    and ',num2str(round(cb_calc_time-60*floor(
    cb_calc_time/60))), 's']);
386
387 %% Functions
388
389 %% CSM and signal chunk computation
390 function [CSM, f, signalChunk] = CSM_computation(signal,
      nChunks, mainLoopIndex, window, noverlap, nfft, fs)
391
392     % Defines Cross Spectral Matrix (CSM), the
    associated frequencies' vector
393     % and the signal chunck relative to the considered
    time sub-interval.
394
395     disp('CSM computation - chunk number ' +
    mainLoopIndex)          % Inform the user about the
    signal chunk (or time sub-interval) being analized
```

```
396
397
398     nMic = size(signal,1);
399     nDataPerChunk = floor(size(signal,2)/nChunks);
            % Number of data gathered by each mic in the
    chunk (time subinterval) considered
400
401     signalChunk = signal( :,      1 + (mainLoopIndex-1)*
    nDataPerChunk   :  mainLoopIndex*nDataPerChunk    );
      % Selecting data of the considered chunk from the
    whole signal
402
403
404     var = mod(nfft,2);           % Reminder of division
    between nfft and 2
405
406     if var == 0                  % This computation gives
407         nFreq = nfft/2+1;            % the number of
    frequencies
408     else                                 % that will be
    saved in the f vector
409         nFreq = (nfft+1)/2;                % depending
    on the nfft parameter and fs
410     end
411
412
413     CSM = zeros(nMic,nMic,nFreq);        % Preallocating
414
415     for ii = 1:nMic                  % For each couple of
     microphones in the array
416         for jj = ii:nMic                 % the following
    computation defines an element in the CSM
417                                              % (actually
    2, as the CSM is a Hermitian matrix)
418             signalx = signalChunk(ii,:);
419             signaly = signalChunk(jj,:);
420
421             [CSM(ii,jj,:),f] = cpsd(signalx,signaly,
    window,noverlap,nfft,fs);
422             CSM(jj,ii,:) = conj(CSM(ii,jj,:)*f(2));
    % (CSM is a Hermitian matrix)
```

```
423         end
424
425          disp(ii);
426     end
427 end
428
429
430 %% Steering vector computation
431 function [PG] = SteeringVector(xn, yn, zn, Array, mirror
    , duct, room, Mvett, Beta, omega, c, N_mic, order,
    Coeff, Source_type, d_dipole)
432
433     PG = zeros(N_mic, 1);
434
435     for j = 1:N_mic
436
437         xm = Array(j,1);
438         ym = Array(j,2);
439         zm = Array(j,3);
440
441         dist_vect = Array(j,:)  -  [xn yn zn];
442
443         delay = (-Mvett*dist_vect'+sqrt((Mvett*dist_vect
    ')^2+Beta^2*norm(dist_vect)^2))/(c*Beta^2);
444
445         dist = sqrt((Mvett*dist_vect')^2+Beta^2*norm(
    dist_vect)^2);
446
447         if Source_type == 1          % Monopole source
448
449             if mirror(1,1) == 1
450
451     %             n_mir = mirror(2:4);
452                 P_mir = mirror(5:7);
453
454                 %
                                    dist_vect_mir = ( Array(j
    ,:) -2*((P_mir-[xn yn zn])*n_mir')*n_mir-[xn yn zn]);
455                 dist_vect_mir = ( Array(j,:) - [xn
    (-2.*P_mir(2)+yn)   zn] );
```

```
456 %                 delay_mir = (-Mvett*dist_vect_mir'  +
     sqrt((Mvett*dist_vect_mir')^2  +  Beta^2*norm(
    dist_vect_mir)^2))   /   (c*Beta^2);
457                 dist_mir = sqrt( (Mvett*dist_vect_mir')
    ^2  +  Beta^2*norm(dist_vect_mir)^2 );
458
459                 PG(j,1) = exp(-1i*omega*delay)/(4*pi()*
    dist) + exp(-1i*omega*dist_mir/c)/(4*pi()*dist_mir);
460
461             elseif duct(1) == 1          % duct(1) = 1
    --> duct on;
462
463                 if duct(2) == 0          % duct(2) = 0
    --> image method off
464
465                     PG(j,1) = GreensFunctionModal_Duct(
    [xn + duct(3)/2 yn + duct(4)/2 zn], xm + duct(3)/2,
    ym + duct(4)/2, zm, Frq(ff), duct(3:4) );
466
467                 elseif duct(2) == 1          % duct(2) =
    1 --> image method on
468
469                     [imgCoord, imgOrder] =
    IMG_computation2D([xn + duct(3)/2, yn + duct(4)/2, zn
    ], duct(3:4));
470
471                     GF = @(r,omega,delay) exp(-1i.*omega
    .*delay)./(4.*pi.*r);
472                     PG(j,1) = GF_computation2D_Moving(GF
    , omega, imgCoord, imgOrder, xm + duct(3)/2, ym +
    duct(4)/2, zm, Coeff, order, Mvett, Beta, c);
473
474                 end
475
476             elseif room(1) == 1            % room(1) = 1
    --> room on;
477
478                 if room(2) == 0            % room(2) = 1
    --> image method on
479
```

```
480                     PG(j,1) = GreensFunctionModal_Room(
    [xn + room(3)/2 yn + room(4)/2 zn + room(5)/2], xm +
    room(3)/2, ym + room(5)/2, zm + room(5)/2, Frq(ff),
    duct(3:5) );
481
482                 elseif room(2) == 1             % room(2)
    = 1 --> image method on
483
484                     [imgCoord, imgOrder] =
    IMG_computation3D([xn + room(3)/2, yn + room(4)/2, zn
     + room(5)/2 ], room(3:5));
485                     % Warning: SG z coord. must be
    smaller than half the Room legnth (lz/2), the mic
    array is in the center of the Room
486
487                     %                       [imgCoord,
    imgOrder] = IMG_computation2D(SG(i,:), Duct(3:4));
488                     GF = @(r,omega,delay) exp(-1i.*omega
    .*delay)./(4.*pi.*r);
489                     PG(j,1) = GF_computation3D_Moving(GF
    , omega, imgCoord, imgOrder, xm + room(3)/2, ym +
    room(4)/2, zm + room(5)/2, Coeff, order, Mvett, Beta,
     c);
490
491                 end
492             else
493
494                 PG(j,1) = exp(-1i*omega*delay)/(4*pi()*
    dist);
495
496             end
497
498         elseif Source_type == 2         % Dipole source
499
500             if mirror(1,1) == 0
501                 PG(j,1) = (1+1i*omega*dist/c)*d_dipole*(
    dist_vect)'*exp(-1i*omega*dist/c)/(4*pi()*dist)^3;
502             else
503                 P_mir = mirror(5:7);
504                 n_mir = mirror(2:4);
```

```
505                 dist_vect_mir = (Array(j,:)  -  2*((
    P_mir - [xn yn zn])*n_mir')*n_mir  -  [xn yn zn] );
506                 dist_mir = norm(dist_vect_mir, 2);
507                 d_dipole_mir = d_dipole-2*(d_dipole*
    n_mir')*n_mir;
508                 PG(j,1) = (1+1i*omega*dist/c)*d_dipole*(
    dist_vect)'*exp(-1i*omega*dist/c)/(4*pi()*dist)
    ^3+(1+1i*omega*dist_mir/c)*d_dipole_mir*(
    dist_vect_mir)'*exp(-1i*omega*dist_mir/c)/(4*pi()*
    dist_mir)^3;
509             end
510         else
511             disp('The source type is not valid')
512         end
513
514     end         % End of for-cycle
515
516 end
```

Bibliography

[1] Jack L.T. Lawrence Antonio J. Torija Rod H. Self. «Psychoacoustic Characterisation of a Small Fixed-pitch Quadcopter». In: *inte.noise* (2019) (cit. on p. 2).

[2] «History of acoustic beamforming». In: 2006 (cit. on p. 3).

[3] Stéphane Moreau Michel Roger. «Tonal-Noise Assessment of Quadrotor-Type UAV Using Source-Mode Expansions». In: *Acoustics* (2020) (cit. on pp. 3, 15).

[4] Guillaume Beaulieu. «Experimental Procedure for Measuring the Noise Annoyance from Multi-Rotor Maneuvers using a UAV System». PhD thesis. University of Liège - School of Engineering and Computer Science, 2022 (cit. on pp. 3, 16).

[5] Renzo Arina. *Fondamenti di Aerodinamica*. Levrotto & Bella, 2015 (cit. on p. 4).

[6] Dean G. Duffy. *Green's functions with applications*. chapman & Hall/CRC, 2001 (cit. on p. 8).

[7] S. Oerlemans. «Detection of aeroacoustic sound sources on aircraft and wind turbines». PhD thesis. University of Twente, 2009 (cit. on p. 9).

[8] Giacomo Gioli Torrione. «Development of a methodology for drone-noise investigation using phased-microphone arrays». PhD thesis. Politecnico di Torino, 2023 (cit. on pp. 14, 19).

[9] Valerio Scollo. «Investigation of beamforming techniquesfor rotating noise sources». PhD thesis. Politecnico di Torino, 2022 (cit. on p. 14).

[10] Roberto Merino-Martinez. «Integration methods for distributed sound sources». In: *International Journal of Aeroacoustics* (2019) (cit. on p. 15).

[11] William Devenport Stewart Glegg. *Aeroacoustics of Low Mach Number Flows*. Academic Press, 2017 (cit. on p. 17).

[12] Riccardo Zamponi. «Development of Acoustic Beamforming Techniques for Accurate Wind-tunnel Tests». PhD thesis. Università Politecnica delle Marche, 2016 (cit. on p. 19).

[13] Ennes Sarradj. «Three-Dimensional Acoustic Source Mapping with Different Beamforming Steering Vector Formulations». In: *Advances in Acoustics and Vibration* (2012) (cit. on p. 21).

[14] *ID2MOVE, official website*. URL: `https://www.id2move.eu/our-services/test-range/` (cit. on p. 22).

[15] *DJI Pantom 3, data sheet*. URL: `https://www.dji.com/it/phantom-3-pro` (cit. on p. 23).

[16] *DJI Air 2S, data sheet*. URL: `https://www.dji.com/air-2s/specs` (cit. on p. 24).

[17] Nicolas Bouanati. «Acoustic beamforming for aeroacoustics». Project Report 2016, von Karman Institute for Fluid Dynamics, 2016 (cit. on p. 26).

[18] Teng Zhou, Hanbo Jiang, and Xiangyu Sun. «Noise source imaging measurements for small-scale multi-propeller systems». In: *Applied Acoustics* (2022) (cit. on p. 26).

[19] *Arqus motion capture camera, Tech Specs*. URL: `https://www.qualisys.com/cameras/arqus/#tech-specs` (cit. on p. 27).

[20] Roberto Merino-Martinez. «Microphone arrays for imaging of aerospace noise sources». PhD thesis. Technische Universiteit Delft, 2018 (cit. on p. 49).