Signal concentration enhancement in the time-frequency domain using adaptive compressive sensing

Volarić, Ivan

UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Ivan Volarić

SIGNAL CONCENTRATION ENHANCEMENT IN THE TIME-FREQUENCY DOMAIN USING ADAPTIVE COMPRESSIVE SENSING

DOCTORAL THESIS

Rijeka, 2017.

UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Ivan Volarić

SIGNAL CONCENTRATION ENHANCEMENT IN THE TIME-FREQUENCY DOMAIN USING ADAPTIVE COMPRESSIVE SENSING

DOCTORAL THESIS

Supervisor: Prof. D. Sc. Viktor Sučić

Rijeka, 2017.

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET

Ivan Volarić

POBOLJŠANJE KONCENTRACIJE SIGNALA U VREMENSKO-FREKVENCIJSKOJ DOMENI PRIMJENOM ADAPTIVNOG KOMPRIMIRANOG UZORKOVANJA

DOKTORSKA DISERTACIJA

Mentor: prof. dr. sc. Viktor Sučić

Rijeka, 2017.

Doctoral thesis supervisor: Prof. D. Sc. Viktor Sučić, Faculty of Engineering, University of Rijeka, Croatia.

The doctoral thesis was defended on 04. 09. 2017. at the Faculty of Engineering of the University of Rijeka. Committee for the defense of the doctoral thesis:

1. Prof. D. Sc. Miroslav Vrankić, Faculty of Engineering, University of Rijeka, Croatia - Committee Chair.
2. Prof. D. Sc. Viktor Sučić, Faculty of Engineering, University of Rijeka, Croatia.
3. Prof. D. Sc. Damir Seršić, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia.

Sveučilište u Rijeci
TEHNIČKI FAKULTET
Fakultetsko vijeće

KLASA: 030-09/16-01/10
URBROJ: 2170-57-01-16-14
Rijeka, 28. listopada 2016.

Fakultetsko vijeće Tehničkog fakulteta Sveučilišta u Rijeci, na svojoj 1. sjednici u ak. god. 2016./17., održanoj 28. listopada 2016., donijelo je sljedeću

O D L U K U

Sukladno izvješću Stručnog povjerenstva za ocjenu teme doktorske disertacije u sastavu: prof. dr. sc. Viktor Sučić (predsjednik), izv. prof. dr. sc. Miroslav Vrankić (član) i doc. dr. sc. Jonatan Lerga (član) utvrđuje se da pristupnik **Ivan Volarić, mag. ing. el.**, ispunjava Zakonom propisane uvjete za izradu doktorske disertacije pod naslovom

Poboljšanje koncentracije signala u vremensko-frekvencijskoj domeni primjenom adaptivnog komprimiranog uzorkovanja.

Mentorom se imenuje prof. dr. sc. Viktora Sučića.

Dekanica

Prof. dr. sc. Jasna Prpić-Oršić

Dostaviti:

1.) Ivan Volarić, mag. ing. el.
2.) Mentor prof. dr. sc. Viktor Sučić
3.) Služba studentske evidencije
4.) Pismohrana FV

You miss 100% of the shots you don't take.
-Wayne Gretzky

Acknowledgments

First of all, I would like to thank my supervisor, Prof. D. Sc. Vikor Sučić for allowing me the space and freedom I needed to finish my PhD research and for the continued guidance and support he has provided from day one. Without him, this thesis would not have been possible.

I would like to thank members of my thesis evaluation committee, Prof. D. Sc. Miroslav Vrankić. and Prof. D. Sc. Damir Seršić for their time and advices. I would also like to thank all the members of staff at Department of Automation and Electronics for their help.

And finally, a special thanks goes to my friends, colleagues and family who endured me over the last few years and helped me keep my sanity.

Author

Abstract

Signals with the time-varying frequency content are best represented in the joint time-frequency domain, with the components instantaneous frequency laws being their key non-stationary features. However, the most commonly used methods for the time-frequency distribution (TFD) calculation generate unwanted artifacts, making their interpretation more difficult. In order to overcome this limitation, a number of methods have been proposed utilizing the compressive sensing (CS) for the artifact removal, while the unavoidable resolution loss is reduced by the signal reconstruction algorithm which, in its core, solves a linear unconstrained optimization problem.

The performance of the existing methodology relies primarily on the user-predefined parameters, namely the CS area and input parameters of the sparse reconstruction algorithm, which are in most cases chosen experimentally. This approach has resulted in the unreliability of the sparse TFD methods, as the parameters which perform well for one signal will not necessarily perform equally for a different signal.

In order to overcome this problem, three adaptive methods are proposed in this thesis, jointly resulting an an adaptive data-driven solution. The first proposed algorithm, adaptively detects the CS area, while the remaining two algorithms are adaptive sparse reconstruction algorithms based on the intersection of confidence intervals rule and the localized Rényi entropy, respectively. The proposed adaptive sparse reconstruction algorithms can be used in the conjunction with the adaptive CS area selection method in order to increase the concentration of the resulting sparse TFD even further. The here-proposed methods are tested on synthetical and real-life signals, and the obtained results are compared with the results obtained with the currently available state-of-the-art sparse TFD reconstruction methods.

Keywords: Time-frequency signal representation, Ambiguity function, Cross-terms suppression, Signal sparsity, Compressive sensing, Linear unconstrained optimization, Intersection of confidence intervals method, Rényi entropy.

Sažetak

U velikom broju praktičnih problema promatranje signala kao funkciju vremena ne pruža dovoljno informacija o samoj prirodi promatrane pojave, te dolazi do potrebe za uvidom u frekvencijski sadržaj signala što je moguće napraviti pomoću Fourierove transformacije. Međutim, Fourierovom transformacijom signala gube se njegove vremenske značajke, te je stoga ovakav pristup neprimjeren za nestacionarne signale, tj. signale čija se frekvencija mijenja kroz vrijeme. Pri analizi nestacionarnih signala poželjno je promatrati energiju signala kao funkciju i vremena i frekvencije istodobno, pa se takav prikaz naziva vremensko-frekvencijska distribucija (VFD) signala. Energija idealne VFD usko je lokalizirana oko trenutne frekvencije komponenti signala, što je u praksi zbog problema opisanih u nastavku teško postići.

Računski najjednostavnije VFD su one linearne, prvenstveno Fourierova transformacija na vremenskom otvoru i Gaborova transformacija. No, linearne VFD imaju ograničenu rezoluciju, te uobičajeno podrazumijevaju kompromis između rezolucije u vremenu i rezolucije u frekvenciji, stoga se u praksi najčešće koriste kvadratne VFD (KVFD). Kada se signal sastoji od više komponenti ili od nelinearne komponente, zbog svoje kvadratne prirode, KVFD uvode neželjene članove, tzv. artefakte ili među-članove, koji se pojavljuju između svakog para komponenti signala te uvelike otežavaju interpretaciju KVFD. Pošto su među-članovi visoko oscilatorni, u domeni neodređenosti, tj. dvodimenzionalnoj Fourierovoj transformaciji VFD, među-članovi su locirani dalje od ishodišta domene, stoga ih je moguće filtrirati s nisko-propusnim filtrom. Međutim, filtriranje na opisani način uklanja i dio korisnih informacija u domeni neodređenosti, tzv. auto-članove, što dovodi do gubitka rezolucije VFD.

Osnovna, nefiltrirana KVFD naziva se Wigner-Villeova distribucija (WVD), a primjenom 2D nisko-propusnog filtra u domeni neodređenosti, moguće je dizajnirati beskonačan broj različitih KVFD s različitim razinama kompromisa između učinkovitosti uklanjanja među-članova i gubitka rezolucije auto-članova. U literaturi među korištenijim KVFD nalaze se: izglađena pseudo WVD, Choi-Williams distribucija, Born-Jordan distribucija, B distribucija, te njene modifikacije. Bolje performanse KVFD mogu se postići nezavisnim filtriranjem po vremenu i po frekvenciji, ali takve metode često zahtijevaju *a priori* poznavanje prirode promatranog signala, što u praksi najčešće nije

slučaj.

Jedna od nedavno predloženih metoda uklanjanja među-članova iz KVFD koristi svojstva prorijeđenosti signala i komprimiranog uzorkovanja (KU). U praksi ustaljene metode uzorkovanja slijede Shannon-Nyquistov teorem, no KU nudi alternativu Shannon-Nyquistovom teoremu, omogućavajući točnu rekonstrukciju signala uzorkovanog ispod Nyquistove frekvencije, uz povećanje matematičke zahtjevnosti algoritma za rekonstrukciju signala. No, da bi rekonstrukcija signala rezultirala smislenim rješenjem promatrani signal mora biti K-prorijeđen, što podrazumijeva da ga je moguće prikazati u nekoj domeni sa K uzoraka, gdje je $K << N_t$, gdje je N_t broj uzoraka u vremenu. Većina signala nije prorijeđena u domeni promatranja, ali signal može postati prorijeđen (ili približno prorijeđen)u nekoj drugoj domeni. Na primjer, sinusoidni signal moguće je prikazati sa samo jednim uzorkom u frekvencijskoj domeni. U vremensko-frekvencijskoj obradi signala navedena svojstva KU mogu se iskoristiti na način da se komprimirano uzorkuju oni uzorci domene neodređenosti za koje se zna da ne sadrže među-članove, tj. oni uzorci koji se nalaze blizu ishodišta domene. Rekonstrukcija VFD je moguća iz razloga što je VFD po svojoj prirodi prorijeđena, tj. sastoji se od trajektorija trenutnih frekvencija komponenti signala, koje se mogu opisati s $N_c N_t << N_f N_t$ uzoraka, gdje su N_c i N_f broj komponenti u signalu i broj uzoraka u frekvenciji.

Učinak postojeće metodologije VFD rekonstrukcije uvelike se oslanja na korisnički predefiniranim parametrima, posebice na području KU i na ulaznim parametrima rekonstrukcijskog algoritma, čiji se odabir vrši eksperimentalnim putem. Takav pristup dovodi do nepouzdanosti VFD rekonstrukcijskih metoda, pošto ulazni parametri koji rezultiraju dobrim rezultatom za jedan signal neće nužno rezultirati jednako dobrim rezultatom za neki drugi signal. U svrhu prevladavanja navedenog problema u ovoj doktorskoj disertaciji predložene su tri adaptivne metode koje će smanjiti potrebu za intervencijom korisnika i time povećati pouzdanost dobivenih rezultata. Prva metoda adaptivno odabire područje KU, dok su preostale dvije metode rekonstrukcijski algoritmi koji se temelje na pravilu presjecišta intervala pouzdanosti i na vremenski lokaliziranoj Rényeovoj entropiji. Predloženi rekonstrukcijski algoritmi mogu su kombinirati s metodom adaptivnog odabira područja KU u svrhu još većeg porasta koncentracije rezultirajuće prorijeđene VFD. Opisani algoritmi testirani su na sintetičkim signalima i signalima iz stvarnog života, te su dobiveni rezultati uspoređeni s rezultatima dobivenim primjenom najnovijih VFD rekonstrukcijskih metoda.

Ključne riječi: Vremensko-frekvencijski prikaz signala, Domena neodređenosti, Filtriranje među-članova, Prorijeđenost signala, Komprimirano uzorkovanje, Bezuvjetna linearna optimizacija, Metoda presjecišta intervala pouzdanosti, Rényeva entropija.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

In many practical problems observing a signal as a function of time does not provide adequate information about the nature of the observed phenomenon, arising the need for an insight into its frequency content, which can be done through the signal's Fourier transformation. However, by calculating a Fourier transformation of the signal, signal time attributes are lost, making this approach inadequate for the non-stationary signals, that is, signals with a time-varying frequency content. The preferable observation domain, when dealing with non-stationary signals, is the joint time-frequency (TF) domain, providing an insight into the signal energy as a function of both time and frequency, simultaneously. Energy of the ideal time-frequency distribution (TFD) is strictly localized around the instantaneous frequency of the individual signal components; however such energy concentration in real-life applications is impossible to accomplish because of the problems described in the sequel.

One of the simplest ways to calculate a TFD is with the linear class of the TFDs, namely the short-time Fourier transformation and the Gabor transformation [24, 15]. However, the TFDs calculated in this way have a limited resolution and usually involve a trade-off between the resolution in time and the resolution in frequency. This is why in practice the quadratic class of the TFDs (QTFD) are more commonly used; however, the main drawback of the QTFD class comes from its quadratic nature, which introduces unwanted artifacts, also called the cross-terms [24, 15]. When the observed signal has more than one component the outer artifacts will appear midway between the each pair of the signal components. On the other hand, when the observed signal has a nonlinear frequency modulated component, the inner artifacts will appear making the TFD interpretation more challenging. Since the cross-terms are highly oscillatory, they can be filtered out by the two dimensional low-pass filter in the ambiguity function (AF); however in doing so, the concentration of the auto-terms gets reduced as well. The original, unfiltered QTFD

is called the Wigner-Ville distribution [80, 76], and one can design an infinite number of different QTFDs with the different trade-off levels between the cross-terms elimination and the auto-terms concentration preservation. In literature among the most commonly used QTFDs are: smoothed pseudo Wigner-Ville distribution [15], Choi-Williams distribution [22], Born-Jordan distribution [49], B distribution [8], and its modifications [39, 16]. Better performance of the QTFDs can be achieved through the separable kernel design in time and frequency domains [15]; however, the performance of such QTFDs is often dependent on the *a priori* knowledge about the nature of the observed signal, which is in practice often unavailable.

The need for a trade-off between the interference suppression and the TFD localization has led to a number of TFD concentration improvement methods. One of the recently proposed methods for the cross-terms removal is based on the compressive sensing [30, 64, 54, 31]. Over the last few years, the compressive sensing (CS) has been an important research topic [21, 27, 20], with applications in medicine [48, 75, 35], geophysics [23, 33], communication [38], etc. The theory behind the compressive sensing provides an alternative to the Shannon-Nyquist sampling theorem, allowing sampling with the sub-Nyquist frequencies; however, the lower sampling frequency comes with the cost of a mathematically significantly more demanding signal reconstruction algorithm. The reconstruction of the CS signal is only possible if a signal is sparse in the certain domain, which means that the signal can be represented in this domain with K nonzero coefficients, where $K << N_t$, N_t being the number of signal samples in the time domain [21, 27]. Most signals are non-sparse in the domain of interest, but can become sparse (or approximately sparse) by applying a domain transformation. For example, a sinusoidal signal can be represented with only one sample in the frequency domain. The sparse signal reconstruction algorithm can be utilized in order to avoid a trade-off between the cross-terms suppression and the auto-terms concentration loss by picking the AF samples corresponding to the auto-terms, while discarding the interfering ones (in a similar way as a low-pass filter does, just more strictly) [30]. The TFD reconstruction is possible because the TFDs are inherently sparse, as they are composed from trajectories describing the instantaneous frequency law of each individual signal component, containing only $N_c N_t << N_f N_t$ samples (N_c and N_f being number of the signal components and number of the available frequency bins, respectively).

The current methods utilizing the CS and the sparsity constraint for the sparse TFD reconstruction heavily rely upon the user predefined parameters, especially the CS-AF area selection and the choice of input parameters of the sparse reconstruction algorithm. They are both generally selected experimentally, decreasing the overall reliability of the sparse TFD reconstruction, since the inputs which perform well for one signal will not necessarily perform equally for a different signal. This fact has served as a motivation behind this thesis, i.e. to decrease the amount of user input in the TFD reconstruction

process by developing adaptive and reliable methods for the TFD reconstruction.

1.2 Objectives of the Thesis

Current procedures for the sparse TFD reconstruction involve experimental selection of the CS-AF area and the sparse reconstruction algorithm input parameters. Both of these play an important role in the obtained quality of the reconstructed sparse TFD, while their experimental selection decreases the overall reliably of the TFD methods based on the sparsity constraint, as they require intervention of the signal specialist in order to select a proper set of inputs for the signal under consideration.

If the selected CS-AF area is too big, the selected AF samples may include undesired information about the cross-terms, which if captured would reappear in the sparse TFD. On the other hand, to few CS-AF samples may significantly degrade the concentration of the resulting sparse TFD; that is if the selected CS-AF samples do not provide enough information about the auto-terms. These facts lead to the conclusion that the ideal CS-AF area should contain all of the samples originating from the auto-terms, and none from the cross-terms. In that case, reconstruction of the TFD would be trivial and would imply taking a two-dimensional Fourier transformation of the selected CS-AF area. In practical applications, where no prior information about the signal is available, we try to capture as much AF samples corresponding to the auto-terms without including the cross-terms. With CS-AF area selected in this way, the requirements of the sparse reconstruction algorithm will be lowered as much as possible, while guarantying complete removal of the cross-terms.

Furthermore, the selection of the proper input parameters of the sparse reconstruction algorithm plays an important role as well. The choice of the algorithm parameters can not be generalized, as it is both algorithm and signal dependent. However, improper selection of the input parameters may result in a slower convergence rate of the sparse reconstruction algorithm or in a lower concentration of the resulting sparse TFD; in the worst case scenario, solution of the sparse reconstruction algorithm may not even converge at all.

Both of the discussed problems have served as a motivation when defining the two main objectives of this thesis, which when combined will increase the overall reliably of the sparse TFD calculation methods and decrease the need for an intervention of the signal specialist:

- Develop a method for an automatic selection of the optimal CS-AF area. The CS-AF area picked by the proposed method will contain as much as possible information about the energy of the auto-terms, with minimal cross-term inclusion.

- Develop multiple algorithms for the sparse TFD reconstruction, resulting in a more reliable and less input dependent approach than the existing approaches.

1.3 Contributions of the Thesis

The original scientific contributions presented in this thesis can be summarized in the following three points:

- Development of the adaptive algorithm for selection of the compressed sensed area in the ambiguity function. The samples of the ambiguity function picked by the here-proposed algorithm ensure the optimal amount of data for the sparse TFD reconstruction, resulting in a better TFD concentration and a faster sparse reconstruction algorithm convergence rate (Section 4.1 and Section 5.1).

- Development of the adaptive sparse TFD reconstruction algorithm based on the fast intersection of confidence intervals rule. The here-proposed fast intersection of confidence intervals rule has been used as a method for the adaptive selection of the threshold value in the sparse TFD reconstruction based on the ℓ_0 and ℓ_1 norm minimization leading to a faster sparse reconstruction algorithm convergence rate (Section 4.2.4 and Section 5.2).

- Development of the adaptive sparse TFD reconstruction algorithm based on the localized Rényi entropy information. The information about the TFD sparsity level obtained through the localized Rényi entropy has been used in order to enhance the concentration of the sparse TFD obtained through the ℓ_0 norm minimization (Section 5.3).

1.4 Organization of the Thesis

The thesis is organized in six chapters and four appendices. The short overview of the thesis chapters is given below.

- **Chapter 1** gives a motivation behind this thesis. It summarizes the main objectives and the original scientific contributions of this thesis. The thesis organization and the short review through thesis chapters are also given.

- **Chapter 2** introduces the basic concepts of the time-frequency signal processing and its advantages over the one dimensional signal processing in time and frequency domain. The concept of the ideal TFD and its properties are discussed, followed by the review of the classical numerical methods for the TFD calculation. Special importance is given to the quadratic class of the TFDs, in particular, the introduction of the cross-terms and classical methods of their removal.

- **Chapter 3** gives a short overview of the compressive sensing and the signal reconstruction based on the sparsity constraints. Application of described methods for the TFD concentration enhancement is described in details, with the special emphasis on the sparse reconstruction algorithms based on the ℓ_0 norm and the ℓ_1 norm minimization. The state-of-the-art sparse reconstruction algorithms are presented, and their performance is tested on synthetical and real-life signals.

- **Chapter 4** introduces three adaptive methods: the adaptive parallelogram kernel, the denoising methods based on supplementing the local polynomial approximation method with the rules derived from the intersection of confidence intervals rule, and the localized Rényi entropy. The methods are reviewed in their original form and applications, while Chapter 5 thoroughly describes their application in the context of the sparse TFD concentration enhancement.

- **Chapter 5** presents the main original scientific contributions of this thesis. It introduces the algorithm for the adaptive selection of the compressive sensing area in the ambiguity function, and two sparse reconstruction algorithms based on the fast intersection of confidence intervals rule and the localized Rényi entropy, respectively. The performance of the here-proposed methods is tested on synthetical and real-life signals, and the obtained results are compared to the results obtained in Chapter 3, corresponding to the state-of-the-art TFD reconstruction methods.

- **Chapter 6** summarizes the key conclusions of this thesis and outlines directions for the future research based on the work and results presented in the thesis.

The methods proposed in this thesis have been implemented in the MATLAB with the time-frequency toolbox[1], and the here-developed algorithms are given in Appendices B - D as MATLAB functions.

[1] Available at: `http://tftb.nongnu.org/`

Chapter 2

Time-Frequency Signal Representations

In this chapter necessity of the joint time-frequency representation and its advantages over the one-dimensional time or frequency signal representation are discussed. Furthermore, the concept of the ideal TFD is introduced, and the properties of the ideal TFD are reviewed.

The classical numerical methods for the TFD calculation, like the short time Fourier transformation, and the Wigner-Ville distribution are introduced. The necessity of the TFD performance evaluation is discussed and importance of the TFD optimization is addressed. In addition, the more complex TFD calculation methods are reviewed, including separable and adaptive kernel design. All of the discussed methods are substantiated by examples on synthetical and real-life signals.

2.1 Time-Frequency Concepts

2.1.1 Motivation

Signals in terms of their frequency content can generally be divided into stationary and non-stationary signals. In the case of stationary signals, that is, signals with the constant frequency content, the one-dimensional signal processing is an adequate analysis tool, providing insight into signal energy in either the time or frequency domain, which are connected through the Fourier transformation, denoted as $\mathcal{F}\{\cdot\}$, and its inverse, denoted

as $\mathcal{F}^{-1}\{\cdot\}$:

$$Z(f) = \mathcal{F}\{z(t)\} = \int_{-\infty}^{\infty} z(t)e^{-j2\pi ft}dt, \tag{2.1a}$$

$$z(t) = \mathcal{F}^{-1}\{Z(f)\} = \int_{-\infty}^{\infty} Z(f)e^{j2\pi ft}df. \tag{2.1b}$$

However, in the case of non-stationary signals, that is, signals with the time-varying frequency content, the one-dimensional signal processing methods are insufficient. Instead, the desirable analysis domain for the non-stationary signals is the joint time-frequency domain, with the components instantaneous frequency laws being their key non-stationary features. The multi-component non-stationary signals are analytically expressed as:

$$z(t) = \sum_{i=1}^{N_c} A_i(t)e^{j\varphi_i(t)}, \tag{2.2}$$

where N_c is the number of non-stationary components, while $A_i(t)$ and $\varphi_i(t)$ are slowly varying amplitude and phase content of the signals i-th component, respectively, By analyzing a TFD one can observe the evolution of signal energy as a function of both time and frequency, providing additional information about the signal nature.

2.1.2 Ideal Time-Frequency Distribution

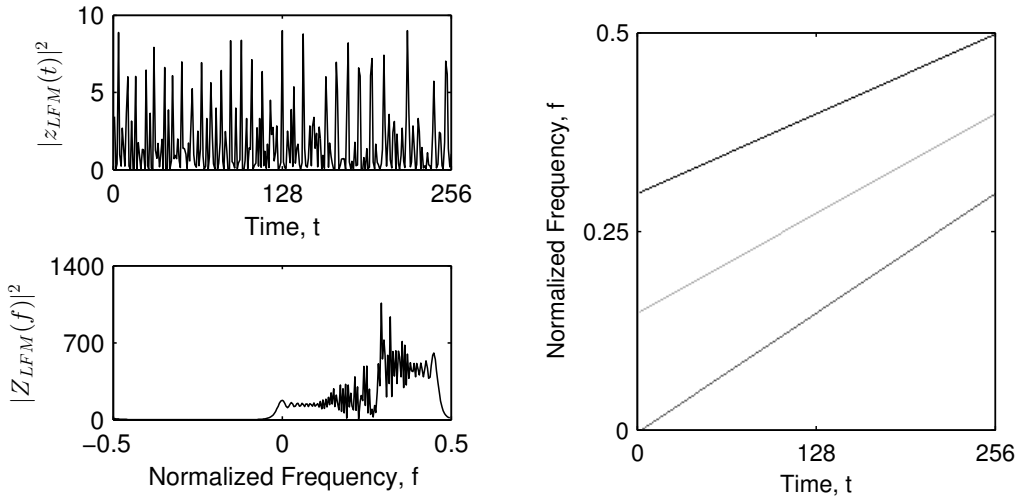
Ideal signal TFD, denoted as $\hat{\rho}_z(t, f)$, is a set of perfectly localized Dirac delta functions:

$$\hat{\rho}_z(t, f) = \sum_{i=1}^{N_c} A_i^2(t)\delta(f - f_{0_i}(t)), \tag{2.3}$$

where $f_{0_i}(t)$ is the instantaneous frequency of the i-th component, calculated as the first derivative of the component phase [24, 15]:

$$f_{0_i}(t) = \frac{1}{2\pi}\frac{d\varphi_i(t)}{dt}. \tag{2.4}$$

Example 2.1. Consider a multi-component non-stationary signal $z_{\text{LFM}}(t) = \sum_{i=1}^{N_c} z_i(t)$ with



(a) (b)

Figure 2.1: Considered three LFM component signal, $z_{\mathrm{LFM}}(t)$: (a) the signal's instantaneous power, $|z_{\mathrm{LFM}}(t)|^2$, and its energy spectrum, $|Z_{\mathrm{LFM}}(f)|^2$, (b) the signal's ideal TFD, $\hat{\rho}_{z_{\mathrm{LFM}}}(t,f)$.

$N_t = 256$ samples, composed of $N_c = 3$ linear frequency modulated (LFM) components:

$$z_1(t) = 0.5e^{j2\pi(0.15t+0.00098t^2)}, \tag{2.5a}$$

$$z_2(t) = e^{j2\pi(0.00117t^2)}, \tag{2.5b}$$

$$z_3(t) = 1.5e^{j2\pi(0.3t+0.00078t^2)}. \tag{2.5c}$$

The signal's instantaneous power, $|z_{\mathrm{LFM}}(t)|^2$, and its energy spectrum, $|Z_{\mathrm{LFM}}(f)|^2$ are shown in Fig. 2.1(a). By observing both of the domains, it can be easily seen that neither representation gives an insight into nature of the considered signal. However, by applying (2.3) and (2.4) to the signal analytical expression (2.5c), the signal's ideal TFD, shown in Fig. 2.1(b), can be calculated as:

$$\begin{aligned}\hat{\rho}_{z_{\mathrm{LFM}}}(t,f) = &\underbrace{0.25\delta(f-(0.15+0.00196t))}_{\hat{\rho}_{z_1}(t,f)} + \\ &+\underbrace{\delta(f-0.00234t)}_{\hat{\rho}_{z_2}(t,f)} + \underbrace{2.25\delta(f-(0.3+1.00156t))}_{\hat{\rho}_{z_3}(t,f)}.\end{aligned} \tag{2.6}$$

The advantages of the joint time-frequency signal representation over the separate time and frequency signal representation are obvious: time and frequency domain do not provide information about the number of signal components, their amplitudes or their durations, while in the joint time-frequency domain all of the signal features are clearly visible.

Example 2.2. Consider a two component non-stationary signal, $z_{\mathrm{nonLFM}}(t) = \sum_{i=1}^{N_c} z_i(t)$,

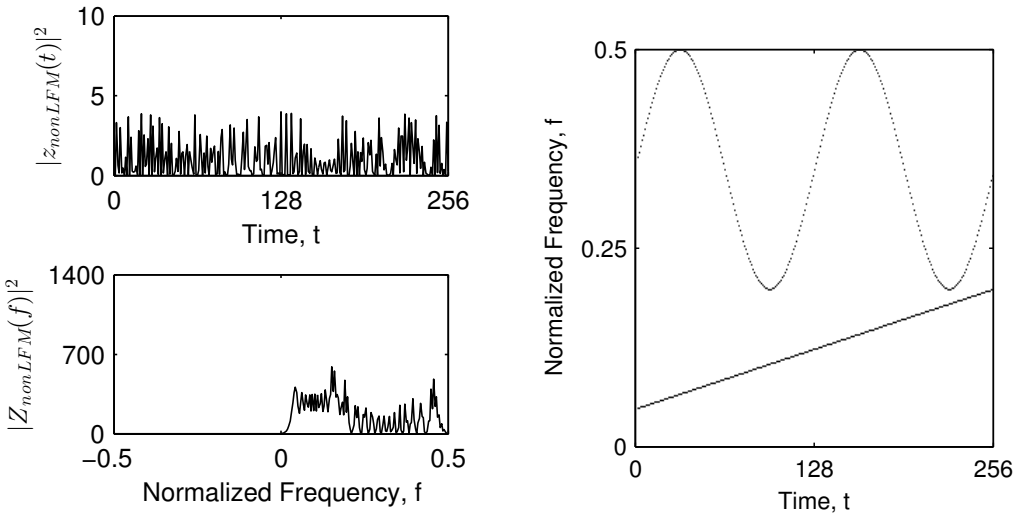


Figure 2.2: Considered two component signal, $z_{\text{nonLFM}}(t)$: (a) the signal's instantaneous power, $|z_{\text{nonLFM}}(t)|^2$, and its energy spectrum, $|Z_{\text{nonLFM}}(f)|^2$, (b) the signal's ideal TFD, $\hat{\rho}_{z_{\text{nonLFM}}}(t,f)$.

composed of the LFM and the sinusoidal frequency modulated component:

$$z_1(t) = e^{j2\pi(0.05t+0.00059t^2)}, \tag{2.7a}$$

$$z_2(t) = e^{j(0.7\pi t+19.2\sin(0.049t-\pi/2))}. \tag{2.7b}$$

The signal's instantaneous power, $|z_{\text{nonLFM}}(t)|^2$, and its energy spectrum, $|Z_{\text{nonLFM}}(f)|^2$ are shown in Fig. 2.2(a), while its ideal TFD is shown in Fig. 2.2(b), and has been calculated as:

$$\hat{\rho}_{z_{\text{nonLFM}}}(t,f) = \underbrace{\delta(f-(0.05+0.00118t))}_{\hat{\rho}_{z_1}(t,f)} + \underbrace{\delta(0.35+0.15\cos(0.049t-\pi/2))}_{\hat{\rho}_{z_2}(t,f)}. \tag{2.8}$$

In most real-life applications, the signal analytical form is impossible to obtain, hence the procedure described in Example 2.1 and Example 2.2 can not be used. Instead, numerical methods for the TFD calculation have been developed, each one with its own advantages and limitations, which will be discussed in the sequel of this chapter.

2.1.3 Properties of Ideal Time-Frequency Distribution

As it can be concluded from (2.3), the TFDs are energy distributions, because of the term $A_i^2(t)$. As an energy distribution, following properties are desirable for TFD to have [24, 15]:

1. **Realness and positivity.** Energy must be real and positive:

$$\hat{\rho}_z(t,f) = \hat{\rho}_z^*(t,f), \tag{2.9a}$$

$$\hat{\rho}_z(t,f) > 0. \tag{2.9b}$$

2. **Time-shift invariance.** A time shift in the signal produces the same time shift in the TFD, that is:

$$z_1(t) = z(t-t_0) \Longrightarrow \hat{\rho}_{z_1}(t,f) = \hat{\rho}_z(t-t_0,f). \tag{2.10}$$

3. **Frequency-shift invariance.** A frequency shift in the signal produces the same frequency shift in the TFD, that is:

$$z_1(t) = z(t)e^{j2\pi f_0 t} \Longrightarrow \hat{\rho}_{z_1}(t,f) = \hat{\rho}_z(t,f-f_0). \tag{2.11}$$

4. **Time marginal.** Instantaneous power, $|z(t)|^2$, is obtained by integration of the TFD w.r.t. frequency:

$$\int_{-\infty}^{\infty} \hat{\rho}_z(t,f)df = |z(t)|^2. \tag{2.12}$$

5. **Frequency marginal.** Energy spectrum, $|Z(f)|^2$, is obtained by integration of the TFD w.r.t. time:

$$\int_{-\infty}^{\infty} \hat{\rho}_z(t,f)dt = |Z(f)|^2. \tag{2.13}$$

6. **Global energy.** Total energy, E_z, is obtained by integration of the TFD over the entire (t,f) plane:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{\rho}_z(t,f)dtdf = \int_{-\infty}^{\infty} |z(t)|^2 dt = \int_{-\infty}^{\infty} |Z(f)|^2 df = E_z. \tag{2.14}$$

7. **Instantaneous frequency.** Instantaneous frequency is calculated as the first moment of the TFD w.r.t. frequency:

$$\frac{\int_{-\infty}^{\infty} f\hat{\rho}_z(t,f)df}{\int_{-\infty}^{\infty} \hat{\rho}_z(t,f)df} = \frac{1}{2\pi}\frac{d}{dt}[\arg(z(t))]. \tag{2.15}$$

8. **Time Delay.** Time delay is calculated as the first moment of the TFD w.r.t. time:

$$\frac{\int_{-\infty}^{\infty} t\hat{\rho}_z(t,f)dt}{\int_{-\infty}^{\infty} \hat{\rho}_z(t,f)dt} = -\frac{1}{2\pi}\frac{d}{df}[\arg(Z(f))]. \tag{2.16}$$

2.1.4 Signal Analytic Associate

It is well known that the frequency spectrum o the real signal, $s(t) \in \mathbb{R}$ will exhibit the Hermitian symmetry, that is:

$$S(-f) = S^*(f), \tag{2.17}$$

thus the negative frequencies can be eliminated without any information loss. The signal without the negative frequencies, denoted as $z(t)$:

$$Z(f) = 0, \quad \text{for} \quad f < 0, \tag{2.18}$$

is an analytic associate of the signal $s(t)$. Note that the generalized multi-component signal, given in (2.2) and the signals $z_{\text{LFM}}(t)$ and $z_{\text{nonLFM}}(t)$ defined in Example 2.1 and Example 2.2 do not contain the negative frequencies, as it is visible in Figs. 2.1(a) and 2.2(a), thus they are analytical associates, and furthermore, all signals in the sequel of this thesis will also be defined in their analytical associate form.

When the signal is defined in the analytical associate form, the mathematical complexity of the numerical TFD calculation is reduced, especially when dealing with the QTFDs, discussed later in Section 2.3, as it eliminates the cross-terms between the positive and the negative frequencies.

The analytic associate, $z(t)$, of the signal $s(t)$ is calculated as [24, 15]:

$$z(t) = s(t) + j\mathcal{H}\{s(t)\}, \tag{2.19}$$

where the operator $\mathcal{H}\{\cdot\}$ denotes the Hilbert transformation:

$$\begin{aligned} \mathcal{H}\{s(t)\} &= \mathcal{F}^{-1}\left\{(-j\,\text{sgn}(f))\mathcal{F}\{s(t)\}\right\} \\ &= s(t) * \frac{1}{\pi t}, \end{aligned} \tag{2.20}$$

where $\mathrm{sgn}(\cdot)$ is a signum function, defined as:

$$\mathrm{sgn}(\xi) = \begin{cases} -1, & \xi < 0, \\ 0, & \xi = 0, \\ 1, & \xi > 0. \end{cases} \tag{2.21}$$

2.2 Linear Time-Frequency Distributions

One of the simplest and most straight-forward numerical methods for the TFD calculation are the linear time-frequency distributions. As the most commonly used representative of this TFD class, the short-time Fourier transformation (STFT) is reviewed in this section. The general idea behind the STFT is to obtain a time dependency by multiplying the signal $z(\tau)$ with the window function $w(\tau)$, centered at a time instance $\tau = t$, and calculate a Fourier transformation at each time instance individually. Analytically, the STFT can be expressed as [24, 15]:

$$\mathrm{STFT}_z(t, f) = \int_{-\infty}^{\infty} z(\tau) w(\tau - t) e^{-j2\pi f \tau} d\tau. \tag{2.22}$$

The key parameter in the STFT calculation is the window length, denoted as Δt, and the window shape. With the window length of $\Delta t = 1$ sample, the STFT will have the best possible time resolution, however the Fourier transformation will produce only 1 sample, thus having the lowest possible frequency resolution. On the other hand, with the window size equal to number of the available samples, $\Delta t = N_t$, the window function becomes $w(\tau) = 1$, and the STFT will be equal to the simple one-dimensional Fourier transformation, having the highest possible frequency resolution, but the lowest possible time resolution. Both discussed window lengths are the worst-case scenarios, each one in its own way; however, any window length between these two extremities, involves a trade-off between the time resolution and the frequency resolution. Larger window will have a lower time resolution, but on the other hand will produce more frequency samples, resulting in a higher frequency resolution, and vice versa, the shorter window will have a smaller time resolution and a higher frequency resolution.

The most commonly used window functions are listed in Table 2.1. The selection of the appropriate window type involves a trade-off between resolving a similar strength components closely located in the frequency domain and resolving a dissimilar strength components distantly located in the frequency domain (i.e. the rectangular window has the best resolution when dealing with the closely located components with similar amplitudes; however, due to its spectral properties, i.e. large side-lobes, it is a poor choice for the more spread-out components with dissimilar amplitudes).

Example 2.3. In this example, the STFT of the three LFM component signal, $z_{\mathrm{LFM}}(t)$, previously defined in Example 2.1 has been calculated with the rectangular window containing $\Delta t = \{9, 49, 199\}$ samples. The STFT which lacks in frequency resolution ($\Delta t = 9$ samples) is shown in Fig. 2.3(a), while on the other hand, the STFT with a poor time resolution ($\Delta t = 199$ samples) is shown in Fig. 2.3(b). Even in the case of the STFT with an optimal window length (selected experimentally as $\Delta t = 49$ samples), shown in Fig. 2.3(c), it can be seen that the STFT lacks in time-frequency resolution when compared to the ideal TFD, shown in Fig. 2.1(b)

Note that the STFT, given by (2.22), does not contain a squared amplitude (this is why it is called a linear distribution), and as such the STFT is not an energy distribution. In order to obtain an energy distribution one can take a squared magnitude of the STFT, also called the spectrogram [24, 15]:

$$\mathrm{SPEC}_z(t,f) = |\mathrm{STFT}_z(t,f)|^2 = \left| \int_{-\infty}^{\infty} z(\tau) w(\tau - t) e^{-j2\pi f \tau} d\tau \right|^2 . \tag{2.23}$$

2.3 Quadratic Time-Frequency Distributions

2.3.1 Wigner-Ville Distribution

To overcome the resolution problems of linear TFDs discussed in Section 2.2, the quadratic TFDs have been developed. Unlike the STFT, the time dependency in QTFDs is obtained by multiplying the signal $z(t)$ with its time-shifted complex conjugated copy, $z^*(t-\tau)$, and taking the Fourier transformation of it. This notation is used in some applications (i.e. radar [47]); however, in the signal processing, the more commonly used notation also time-shifts the starting signal, $z(t+\tau/2)$, and multiplies it with $z^*(t-\tau/2)$. Analytically, this can be expressed as [24, 15]:

$$W_z(t,f) = \int_{-\infty}^{\infty} R_z(t,\tau) e^{-j2\pi f \tau} d\tau, \tag{2.24}$$

where $R_z(t,\tau)$ is the signal localized autocorrelation function (LACF), calculated as:

$$R_z(t,\tau) = z\left(t + \frac{\tau}{2}\right) z^*\left(t - \frac{\tau}{2}\right). \tag{2.25}$$

The $W_z(t,f)$ is also known as the Wigner-Ville distribution (WVD) [76, 80], and provides a perfect localization for a mono-component LFM signal [24, 15].

Table 2.1: The most commonly used window functions, $w(\tau)$.

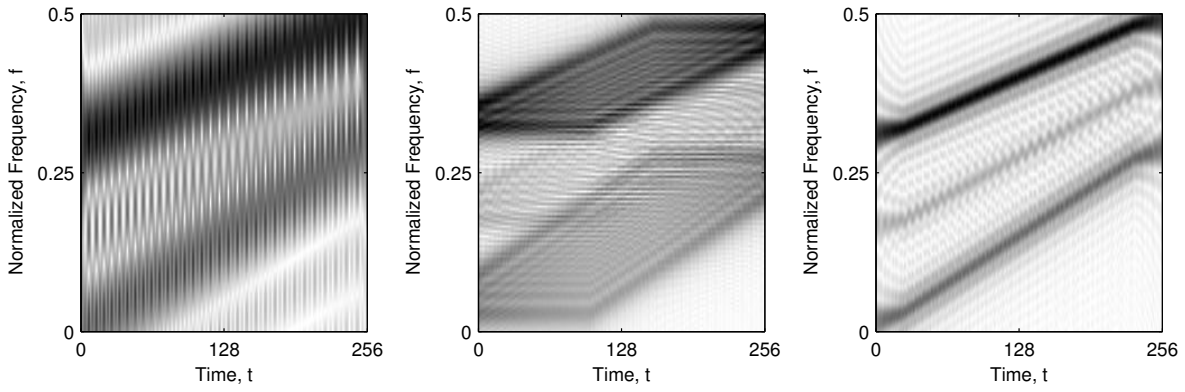
Window name	Window function, $w(\tau)$	
Rectangular	1	$\lvert\tau\rvert < \Delta/2$
Triangular (Bartlett)	$1 - \frac{2\lvert\tau\rvert}{\Delta}$	$\lvert\tau\rvert < \Delta/2$
Hann(ing)	$\frac{1}{2}\left[1 + \cos\left(\frac{\tau\pi}{\Delta}\right)\right]$	$\lvert\tau\rvert < \Delta$
Hamming	$0.54 + 0.46\cos\left(\frac{\tau\pi}{\Delta}\right)$	$\lvert\tau\rvert < \Delta$
Blackman	$0.42 + 0.5\cos\left(\frac{\tau\pi}{\Delta}\right) + 0.08\cos\left(\frac{2\tau\pi}{\Delta}\right)$	$\lvert\tau\rvert < \Delta$
Gaussian	$e^{-\tau^2/\alpha^2}$	$\lvert\tau\rvert < \Delta/2$
Kaiser*	$\frac{I_0\left[\beta\sqrt{1-(\tau/\Delta)^2}\right]}{I_0(\beta)}$	$\lvert\tau\rvert < \Delta$

*I_0 is the zeroth-order modified Bessel function of the first kind, calculated as:

$$I_\alpha(x) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(m+\alpha+1)} \left(\frac{x}{2}\right)^{2m+\alpha},$$

where Γ is the Gamma function, an extension of the factorial function for all complex numbers except for the non-positive integers, calculated as:

$$\Gamma(t) = \int_0^{\infty} x^{t-1} e^{-x} dx.$$



(a) (b) (c)

Figure 2.3: STFT of the three LFM component signal, $z_{\mathrm{LFM}}(t)$, with the window length: (a) $\Delta t = 9$ samples, (b) $\Delta t = 199$ samples, (c) $\Delta t = 49$ samples.

2.3.2 Time-Frequency Related Domains

By inspecting (2.24) it can be concluded that the time-frequency domain is obtained by taking a Fourier transformation of the time-lag domain w.r.t. lag. In addition to the time-frequency domain and the time-lag domain, one can define the doppler-frequency domain and the doppler-lag domain by taking the appropriate Fourier transformations. The doppler-frequency domain, also called the spectral autocorrelation function (SAF), $r_z(\nu, f)$, is calculated by taking a Fourier transformation of the time-frequency domain w.r.t. time:

$$r_z(\nu, f) = \int_{-\infty}^{\infty} W_z(t, f) e^{-j2\pi\nu t} dt, \tag{2.26}$$

while the doppler-lag domain, also called the ambiguity function (AF), $A_z(\nu, \tau)$, is calculated by taking a Fourier transformation of the time-lag domain w.r.t. time:

$$A_z(\nu, \tau) = \int_{-\infty}^{\infty} R_z(t, \tau) e^{-j2\pi\nu t} dt. \tag{2.27}$$

The relationship between the all four time-frequency related domains is shown in Fig. 2.4 by representing the Fourier transformation with the arrows. It is interesting to note that the relationship between the time-frequency domain and the doppler-lag domain is similar to the relationship between the one-dimensional time domain and frequency domain; the slow varying terms in the time-frequency domain are located near the doppler-lag domain origin, and vice versa, the fast varying terms are located further away from its origin.

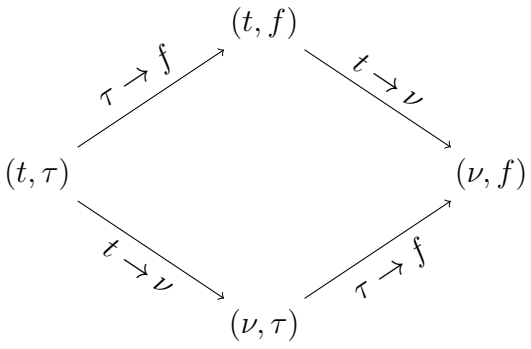


Figure 2.4: Relationship between the four time-frequency related domains. Arrows represent direction of the Fourier transformation.

2.3.3 Multi-component Signals and Cross-terms

The drawback of the WVD takes place when the considered signal has more then one component. In that case, the signal LACF, given by (2.25), expands into:

$$R_z(t,\tau) = \underbrace{\sum_{i=1}^{N_c} z_i\left(t+\frac{\tau}{2}\right) z_i^*\left(t-\frac{\tau}{2}\right)}_{\text{auto-terms}} + \underbrace{\sum_{i=1}^{N_c}\left[z_i\left(t+\frac{\tau}{2}\right)\sum_{\substack{j=1\\ j\neq i}}^{N_c} z_j^*\left(t-\frac{\tau}{2}\right)\right]}_{\text{cross-terms}}, \tag{2.28}$$

while the signal WVD, following the quadratic superposition rule, becomes:

$$W_z(t,f) = \underbrace{\sum_{i=1}^{N_c} W_{z_i}(t,f)}_{\text{auto-terms}} + \underbrace{2\sum_{i=1}^{N_c}\sum_{\substack{j=1\\ j\neq i}}^{N_c} \mathrm{Re}\left\{W_{z_{i,j}}(t,f)\right\}}_{\text{cross-terms}}, \tag{2.29}$$

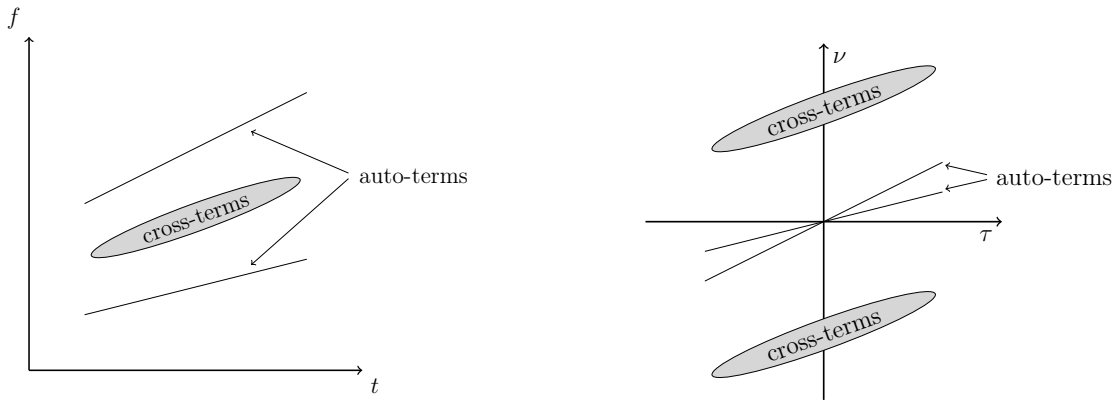
where $W_{z_i}(t,f)$ is the WVD of the signal i-th component, while $W_{z_{i,j}}(t,f)$ is the cross-WVD between the signal i-th and the signal j-th component. Because of this, $W_{z_i}(t,f)$ terms are commonly called the auto-terms, while $W_{z_{i,j}}(t,f)$ terms are commonly called the cross-terms. The auto-terms have the identical form as (2.25) and they provide useful information about the components energy. However, as it can be seen from (2.28), the cross-terms are mathematical byproduct introduced by the LACF's quadratic nature, and will appear midway between the each pair of components, as shown in Fig. 2.5(a). Furthermore, the cross-terms oscillate in the time-frequency domain proportional to distance between the interfering components and have an oscillation direction which is orthogonal to the straight line connecting the components in question [24, 15].

2.3.4 Ambiguity Function Low-Pass Filtering

Because of their oscillatory nature, the cross-terms are located away from the AF domain origin with distance equal to the time-frequency distance between the interfering components [24, 15], as shown in Fig. 2.5(b). Because of their location, the cross-terms can be easily filtered out with a two-dimensional low-pass filter. However, in doing so, the auto-terms get partially filtered out as well, thus reducing the concentration of the signal components in the time-frequency domain. This filtering problem leads to a trade-off between the cross-terms suppression and the auto-terms concentration and consequently leads to a formulation of the quadratic class of the TFDs as [24, 15]:

$$\mathcal{A}_z(\nu,\tau) = A_z(\nu,\tau)g(\nu,\tau), \tag{2.30a}$$

$$\rho_z(t,f) = W_z(t,f) \underset{t}{*}\underset{f}{*} \gamma(t,f), \tag{2.30b}$$



(a) (b)

Figure 2.5: Location of the cross-terms and the auto-terms in the: (a) time-frequency domain, (b) doppler-lag domain.

Table 2.2: Selected TFD kernels in the time-lag domain, the doppler-lag domain, and the doppler-frequency domain.

Distribution	Kernel		
	$G(t,\tau)$	$g(\nu,\tau)$	$\mathcal{G}(\nu,f)$
WVD [76, 80]	$\delta(t)$	1	$\delta(f)$
Levin [51] (Margenau-Hill)	$\frac{1}{2}\left[\delta\left(t+\frac{\tau}{2}\right)+\delta\left(t-\frac{\tau}{2}\right)\right]$	$\cos\left(\pi\nu\tau\right)$	$\frac{1}{2}\left[\delta\left(f+\frac{\nu}{2}\right)+\delta\left(f-\frac{\nu}{2}\right)\right]$
Born-Jordan [49]	$\frac{1}{\lvert 2\alpha_{\text{BJ}}\tau\rvert}\operatorname{rect}\left(\frac{t}{2\alpha_{\text{BJ}}\tau}\right)$	$\operatorname{sinc}\left(2\alpha_{\text{BJ}}\nu\tau\right)$	$\frac{1}{\lvert 2\alpha_{\text{BJ}}\nu\rvert}\operatorname{rect}\left(\frac{f}{2\alpha_{\text{BJ}}\nu}\right)$
Sinc [67]	$\frac{\tau}{\alpha_{\text{SINC}}}\operatorname{sinc}\left(\frac{t\tau}{\alpha_{\text{SINC}}}\right)$	$\operatorname{rect}\left(\frac{\nu\tau}{\alpha_{\text{SINC}}}\right)$	$\frac{\nu}{\alpha_{\text{SINC}}}\operatorname{sinc}\left(\frac{f\nu}{\alpha_{\text{SINC}}}\right)$
Rihaczek [60]	$\delta\left(t-\frac{\tau}{2}\right)$	$e^{-j\pi\nu\tau}$	$\delta\left(f+\frac{\nu}{2}\right)$
Page [55]	$\delta\left(t-\frac{\lvert\tau\rvert}{2}\right)$	$e^{-j\pi\nu\lvert\tau\rvert}$	$\frac{1}{2}\left[\delta\left(f+\frac{\nu}{2}\right)+\delta\left(f-\frac{\nu}{2}\right)\right]$
Choi-Williams [22]	$\frac{\sqrt{\pi\sigma_{\text{CW}}}}{\lvert\tau\rvert}e^{-\pi^2\sigma_{\text{CW}}t^2/\tau^2}$	$e^{-\nu^2\tau^2/\sigma_{\text{CW}}}$	$\frac{\sqrt{\pi\sigma_{\text{CW}}}}{\lvert\nu\rvert}e^{-\pi^2\sigma_{\text{CW}}f^2/\nu^2}$
B [8]	$\lvert\tau\rvert^{\beta_{\text{B}}}\cosh^{-2\beta_{\text{B}}}t$	$\frac{\lvert\tau\rvert^{\beta_{\text{B}}}\lvert\Gamma(\beta_{\text{B}}+j\pi\nu)\rvert^2}{2^{1-2\beta_{\text{B}}}\Gamma(2\beta_{\text{B}})}$	$\frac{-\sin(\pi\beta_{\text{B}}/2)\Gamma(\beta_{\text{B}}+1)}{2^{1-\beta_{\text{B}}}\Gamma(2\beta_{\text{B}})}\frac{\Gamma(\beta_{\text{B}}+j\pi\nu)\rvert^2}{\lvert\pi f\rvert^{\beta_{\text{B}}+1}}$
Modified B [39]	$\frac{\cosh^{-2\beta_{\text{MB}}}t}{\int_{-\infty}^{\infty}\cosh^{-2\beta_{\text{MB}}}\xi d\xi}$	$\frac{\lvert\Gamma(\beta_{\text{MB}}+j\pi\nu)\rvert^2}{\Gamma^2(\beta_{\text{MB}})}$	$\frac{\lvert\Gamma(\beta_{\text{MB}}+j\pi\nu)\rvert^2}{\Gamma^2(\beta_{\text{MB}})}\delta(f)$
Extended modified B [16]	$\frac{\cosh^{-2\beta_{\text{EMB}}}t}{\int_{-\infty}^{\infty}\cosh^{-2\beta_{\text{EMB}}}\xi d\xi}\frac{\lvert\Gamma(\alpha_{\text{EMB}}+j\pi\tau)\rvert^2}{\Gamma^2(\alpha_{\text{EMB}})}$	$\frac{\lvert\Gamma(\beta_{\text{EMB}}+j\pi\nu)\rvert^2}{\Gamma^2(\beta_{\text{EMB}})}\frac{\lvert\Gamma(\alpha_{\text{EMB}}+j\pi\tau)\rvert^2}{\Gamma^2(\alpha_{\text{EMB}})}$	$\frac{\cosh^{-2\alpha_{\text{EMB}}}f}{\int_{-\infty}^{\infty}\cosh^{-2\alpha_{\text{EMB}}}\xi d\xi}\frac{\lvert\Gamma(\beta_{\text{EMB}}+j\pi\nu)\rvert^2}{\Gamma^2(\beta_{\text{EMB}})}$

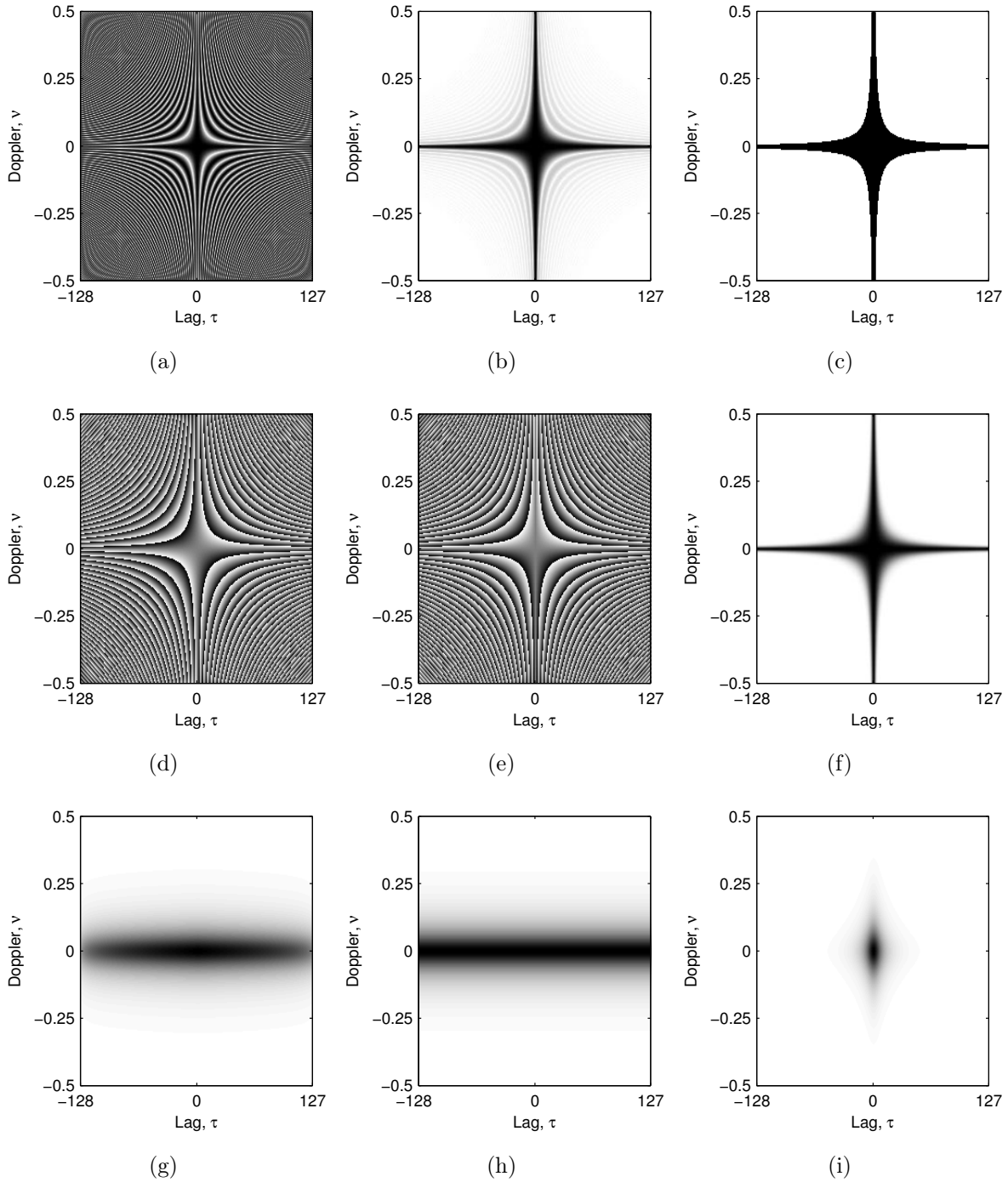


Figure 2.6: Selected TFD kernels in the doppler-lag domain: (a) magnitude of the Levin kernel, (b) magnitude of the Born-Jordan kernel ($\alpha_{\text{BJ}} = 0.4$) (c) magnitude of the Sinc kernel ($\alpha_{\text{SINC}} = 2$), (d) phase of the Rihaczek kernel, (e) phase of the Page kernel, (f) magnitude of the Choi-Williams kernel ($\sigma_{\text{CW}} = 1$), (g) magnitude of the B kernel ($\beta_{\text{B}} = 0.2$), (h) magnitude of the modified B kernel ($\beta_{\text{MB}} = 0.2$), (i) magnitude of the extended modified B kernel ($\beta_{\text{EMB}} = 0.25, \alpha_{\text{EMB}} = 0.1$).

where $g(\nu,\tau)$ and $\gamma(t,f)$ are filter functions (also known as kernels), while $\mathcal{A}_z(\nu,\tau)$ and $\rho_z(t,f)$ are the filtered AF and and the filtered WVD, respectively. One can design an infinite number of various QTFDs with the various levels of trade-off between the cross-terms suppression and the auto-terms concentration. The most commonly used kernels with their respective functions in the time-lag domain, the doppler-lag domain, and the doppler-frequency domain are listed in Table 2.2, and shown in Fig. 2.6.

Example 2.4. In this example, the selected QTFDs of the three LFM component signal, $z_{\mathrm{LFM}}(t)$, previously defined in Example 2.1, have been calculated. The AF of the considered signal is shown in Fig 2.7(a), while the resulting QTFDs are shown in Fig. 2.8. When compared to the optimal STFT, shown in Fig. 2.3(c), the QTFDs higher resolution can easily be noticed; however, the QTFDs are corrupted with the cross-terms between the signal components. Note that the first component, $z_1(t)$, has been modeled in such a way that it is located midway between the remaining two components, $z_2(t)$ and $z_3(t)$, thus overlaying it with the cross-terms between them, making the first component almost unidentifiable in the WVD, shown in Fig. 2.8(a). From the visual inspection of Fig. 2.8, it can be concluded that the best performance among the selected QTFDs has the modified B distribution, shown in Fig. 2.8(h), since the IF laws of the signal components have a slow slew rate, thus in the AF, the auto-terms are located along the lag-axis, fitting into the pass-band of the modified B kernel, shown in 2.6(g).

Example 2.5. In this example, the selected QTFDs of the two component signal, $z_{\mathrm{nonLFM}}(t)$, previously defined in Example 2.2, have been calculated. The AF of the considered signal is shown in Fig. 2.7(b), while the resulting QTFDs are shown in Fig. 2.9. Notice that the cross-terms in WVD, shown in Fig. 2.9(a), are not only located midway between the components, but they are also located around the sinusoidal frequency modulated component. This can be explained by a fact that each TFD point can be considered as a single component, and the cross-terms will appear between all of them. In case of the

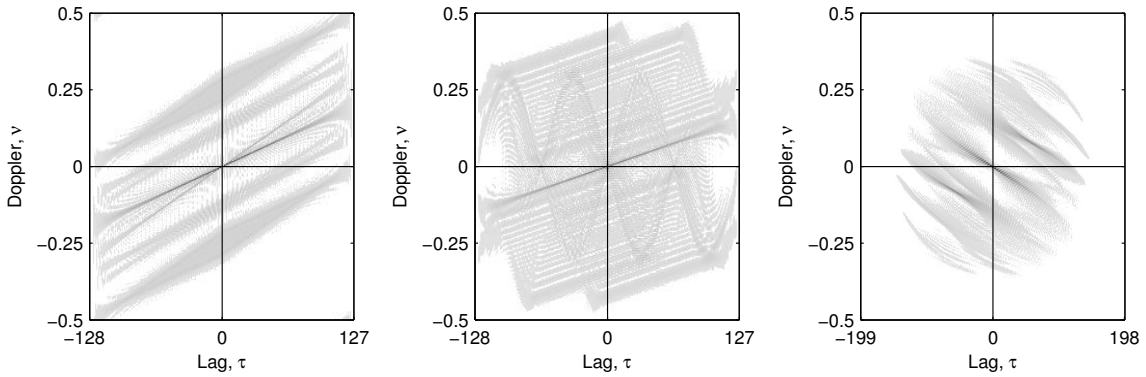


Figure 2.7: Ambiguity functions of the considered signals: (a) $z_{\mathrm{LFM}}(t)$ defined in Example 2.4, (b) $z_{\mathrm{nonLFM}}(t)$ defined in Example 2.5, (c) $z_{\mathrm{bat}}(t)$ defined in Example 2.6.

LFM, all of the cross-terms will align with the auto-terms, forming a single continuous line; however, in case of the non-LFM component, this is not a case, and the cross-terms appear midway between the each pair of the auto-term points. This is why the cross-terms are often divided into the inner-terms (cross-terms which are caused by the components non-linear frequency modulation), and the outer-terms (cross-terms between the components). By visual inspection of Fig. 2.9, the best performance among the selected QTFDs has the extended modified B distribution, shown in Fig. 2.9(a), since the two parameters, α_{EMB} and β_{EMB}, provide the independent user control over the kernel spread along the lag and the doppler axis.

Example 2.6. In this example, the selected QTFDs of the real-life bat (Large Brown Bat, *Eptesicus Fuscus*) echolocation signal[1], $z_{\text{bat}}(t)$, with $N_t = 400$ samples (signal duration is $t = 2.5$ ms, with a sampling period $T_s = 4$ μs) have been calculated. The AF of the considered signal is shown in Fig. 2.7(c), while the resulting QTFDs are shown in Fig. 2.10. The considered signal is composed from three non-LFM components, and it is commonly used as a benchmark in the time-frequency signal processing. As it was the case in Example 2.4 and Example 2.5, by visual inspection of Fig. 2.10, the best performing QTFDs are the modified B distribution, shown in Fig. 2.10(h), and the extended modified B distribution, shown in Fig. 2.10(i).

2.3.5 Separable Kernel Design

A simple way to design a TFD kernel is to split a two-dimensional kernel into two separable one-dimensional kernels:

$$\mathcal{A}_z(\nu,\tau) = G_1(\nu)A_z(\nu,\tau)g_2(\tau), \tag{2.31a}$$

$$\rho_z(t,f) = g_1(t) \underset{t}{*} W_z(t,f) \underset{f}{*} G_2(f). \tag{2.31b}$$

Special cases of the separable kernels are the doppler-independent kernels, with $G_1(\nu) = 1$, that is with $g_1(t) = \delta(t)$, and the lag-independent kernels, with $g_2(\tau) = 1$, that is with $G_2(f) = \delta(f)$. Kernel of the extended modified B distribution [16], shown in Fig. 2.6(i), is an example of a separable kernel, while the kernel of the modified B distribution [39], shown in Fig. 2.6(h), is an example of a lag-independent kernel. On the other hand, the WVD with $g(\nu,\tau) = 1$, has both a lag-independent and a doppler-independent kernel. As discussed in Example 2.5, the separable kernel design provides an independent user control over the kernel spread along the lag and the doppler axis. Because of this, with a proper selection of the lag and the doppler kernel parameters, a user can control the kernel

[1] Available at: `http://dsp.rice.edu/sites/dsp.rice.edu/files/software/batsignal.zip`. The author wishes to thank Curtis Condon, Ken White, and Al Feng of the Beckman Institute of the University of Illinois for the bat data and for permission to use it in this thesis.

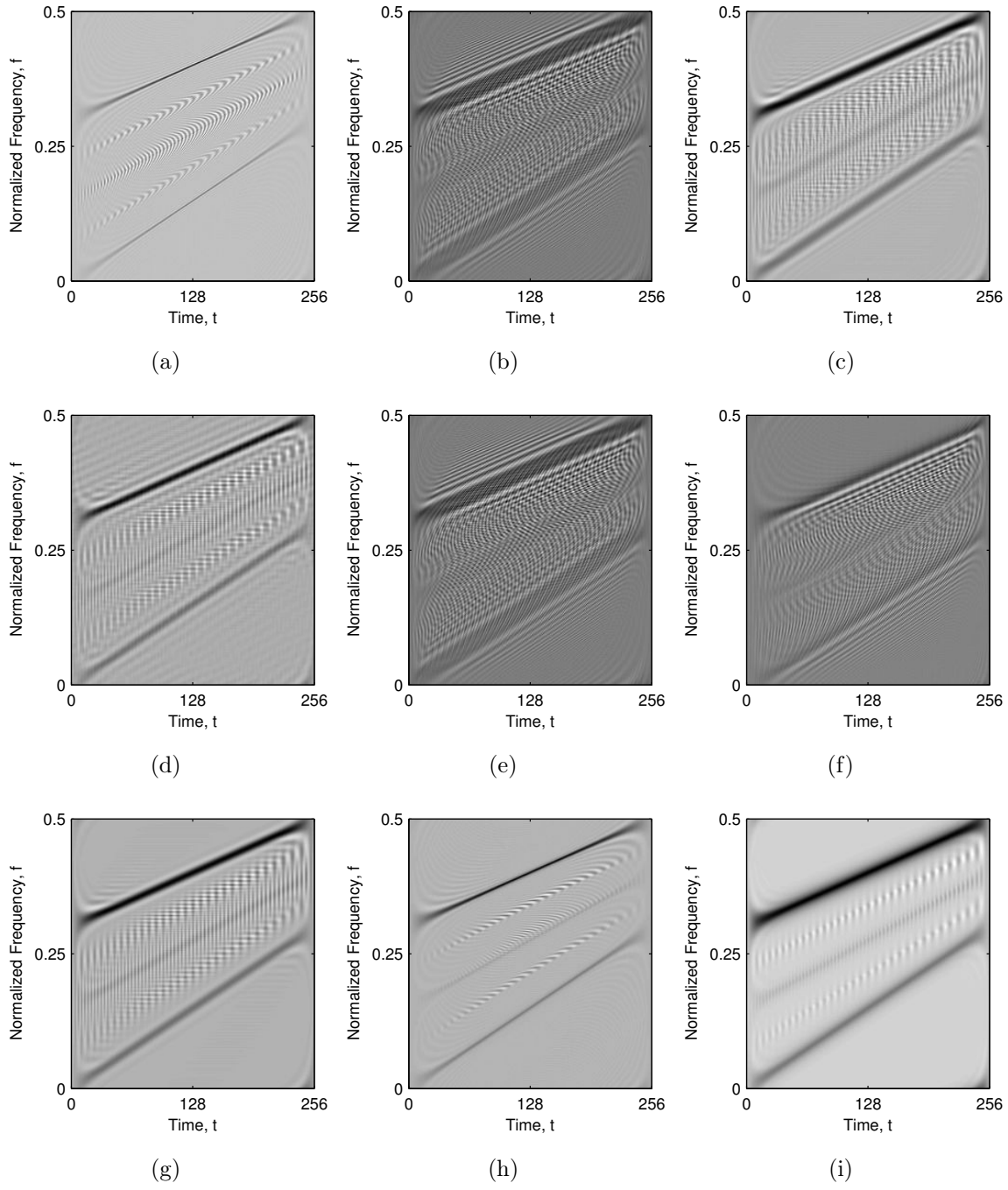


Figure 2.8: Selected QTFDs of the three LFM component signal, $z_{\mathrm{LFM}}(t)$: (a) WVD, (b) Levin distribution, (c) Born-Jordan distribution ($\alpha_{\mathrm{BJ}} = 0.4$), (d) Sinc distribution ($\alpha_{\mathrm{SINC}} = 2$), (e) Rihaczek distribution, (f) Page distribution, (g) Choi-Williams distribution ($\sigma_{\mathrm{CW}} = 1$), (h) Modified B distribution ($\beta_{\mathrm{MB}} = 0.2$), (i) Extended modified B distribution ($\beta_{\mathrm{EMB}} = 0.25, \alpha_{\mathrm{EMB}} = 0.1$).

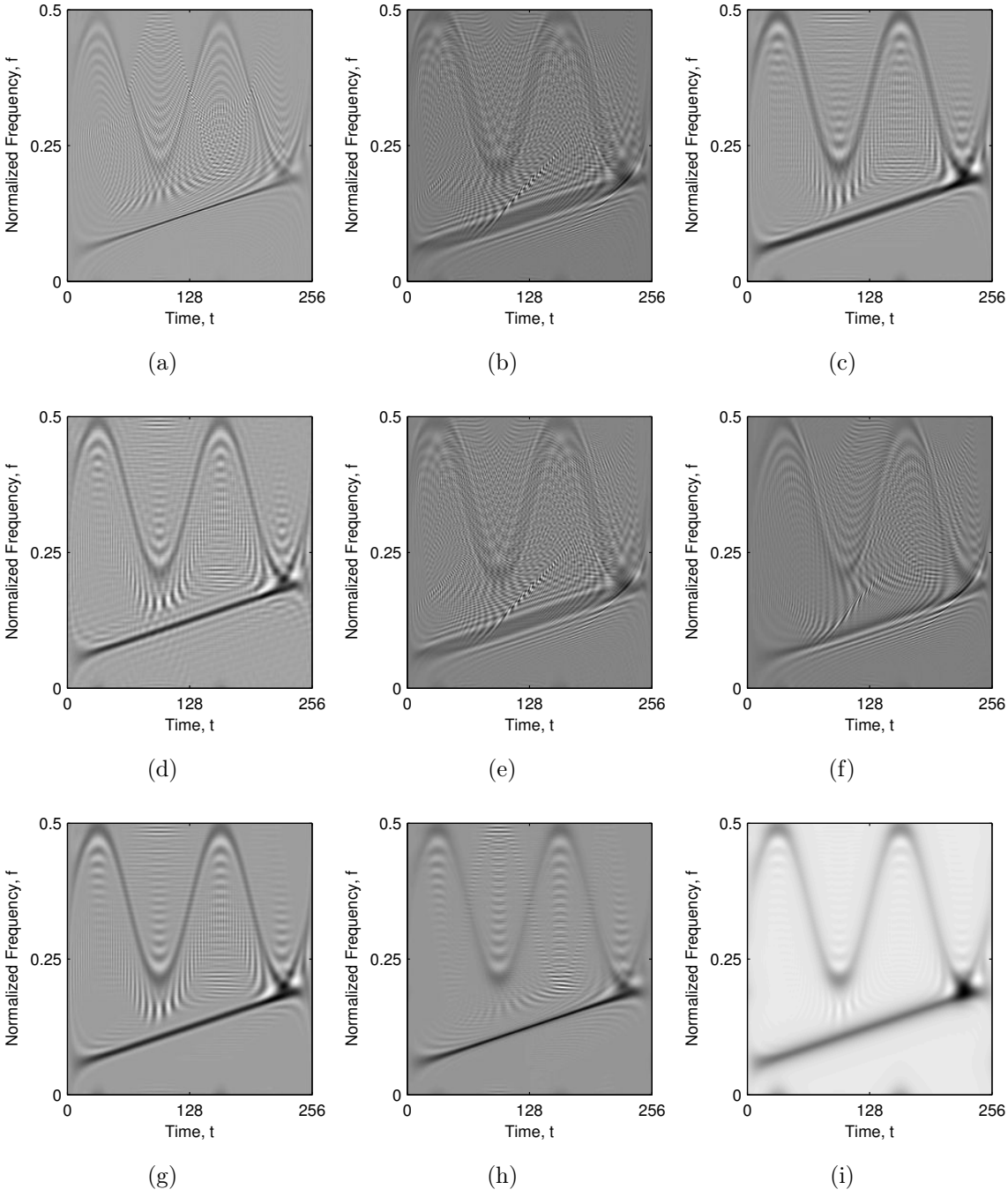


Figure 2.9: Selected QTFDs of the two component signal $z_{\mathrm{nonLFM}}(t)$: (a) WVD, (b) Levin distribution, (c) Born-Jordan distribution ($\alpha_{\mathrm{BJ}} = 0.4$), (d) Sinc distribution ($\alpha_{\mathrm{SINC}} = 2$), (e) Rihaczek distribution, (f) Page distribution, (g) Choi-Williams distribution ($\sigma_{\mathrm{CW}} = 1$), (h) Modified B distribution ($\beta_{\mathrm{MB}} = 0.2$), (i) Extended modified B distribution ($\beta_{\mathrm{EMB}} = 0.25, \alpha_{\mathrm{EMB}} = 0.1$).

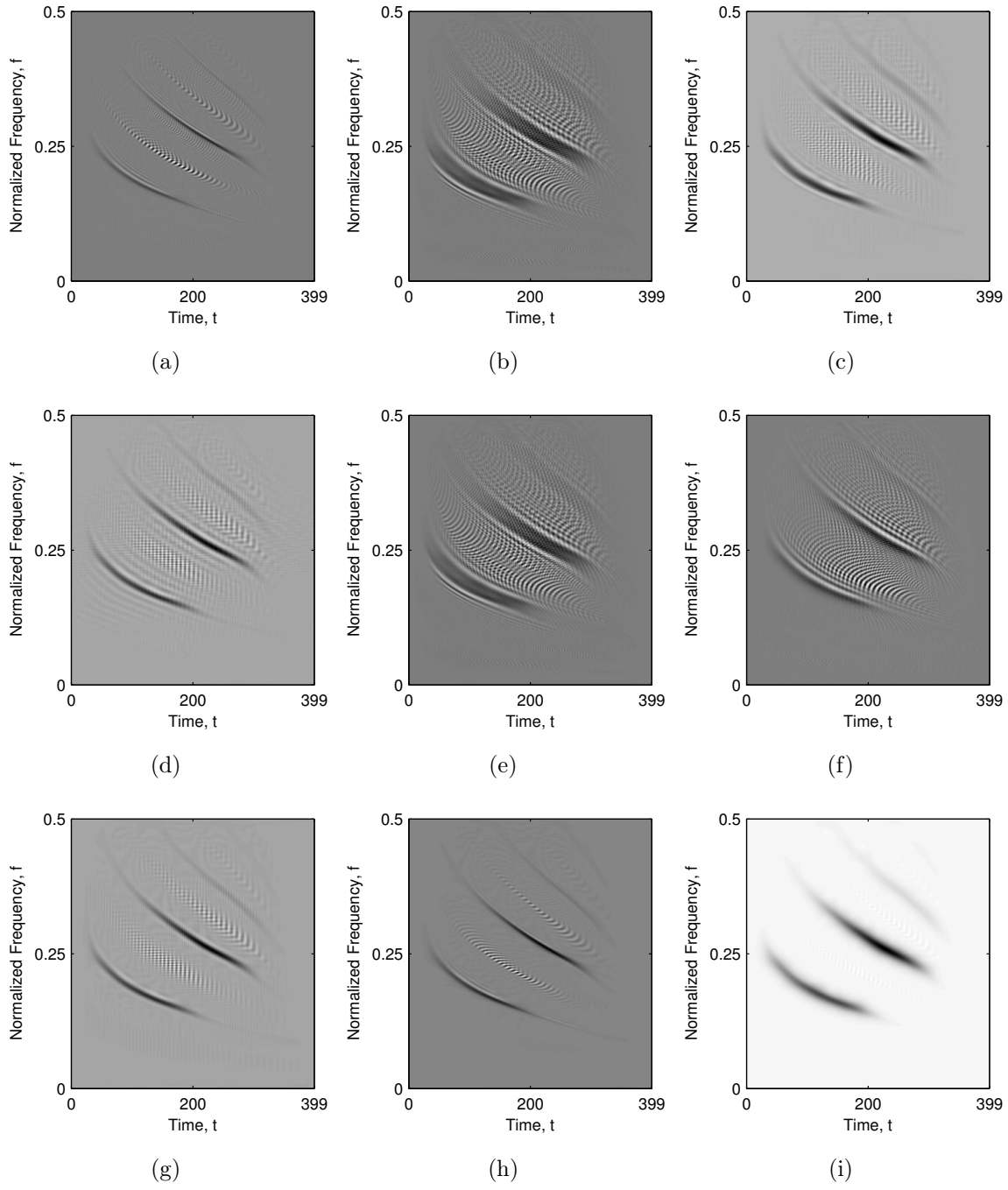


Figure 2.10: Selected QTFDs of the bat echolocation signal, $z_{\mathrm{bat}}(t)$: (a) WVD, (b) Levin distribution, (c) Born-Jordan distribution ($\alpha_{\mathrm{BJ}} = 0.4$), (d) Sinc distribution ($\alpha_{\mathrm{SINC}} = 2.5$), (e) Rihaczek distribution, (f) Page distribution, (g) Choi-Williams distribution ($\sigma_{\mathrm{CW}} = 2$), (h) Modified B distribution ($\beta_{\mathrm{MB}} = 0.2$), (i) Extended modified B distribution ($\beta_{\mathrm{EMB}} = 0.07, \alpha_{\mathrm{EMB}} = 0.1$).

shape in the AF to include only the auto-terms, while avoiding the cross-terms. However, the downside of this approach is the need to have an extensive *a priori* knowledge about the nature of the considered signal in order to correctly design a TFD kernel. A separable two-dimensional TFD kernel can be designed by combining a two one-dimensional window functions, listed in Table 2.2, one for the lag filtering, and one for the doppler filtering.

2.4 Adaptive Kernel Design

2.4.1 TFD Performance Assessment

As discussed previously, when a signal has the *a priori* known behaviour, it is possible to design a TFD kernel which would perfectly capture all of the auto-terms without inclusion of the cross-terms. However, kernel designed in such a way would have perfect performance only for that signal, or very similar class of signals. Because there is no single best performing kernel for all possible applications, a need for the objective TFD performance assessment has arisen. One way of doing it, is to measure the concentration of the auto-terms, which can be done through a calculation of the TFD complexity. Jones and Park [40] proposed the concentration measure, denoted as M_z^{JP}, calculated as a fourth power of ratio between the TFD ℓ_4 norm and the TFD ℓ_2 norm, also known as kurtosis:

$$M_z^{\mathrm{JP}} = \frac{||\rho_z(t,f)||_4^4}{||\rho_z(t,f)||_2^4}, \tag{2.32}$$

where $||\rho_z(t,f)||_p$ denotes the TFD ℓ_p norm, calculated as:

$$||\rho_z(t,f)||_p = \left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\rho_z(t,f)|^p \, dtdf\right]^{\frac{1}{p}}. \tag{2.33}$$

A higher value of the Jones-Park concentration measure indicates the higher TFD concentration, and vice versa. However, if the signal components have various amplitudes, which is the case in many real-life applications, the measure can produce high values if the high amplitude components are highly concentrated, while the low amplitude components are not. This problem can be reduced by calculating the Jones-Park concentration measure over a smaller, localized time-frequency region [40]:

$$M_z^{\mathrm{JPL}}(t_0,f_0) = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Q^2(t-t_0,f-f_0)\rho_z^4(t,f)dtdf}{\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Q(t-t_0,f-f_0)\rho_z^2(t,f)dtdf\right]^2}, \tag{2.34}$$

where $Q(t,f)$ is the localization weighting function used to select a measurement region. However, the downside of the method is its high dependance on a proper $Q(t,f)$ selection and its computational requirements.

Stanković [68] proposed an alternative concentration measure for the normalized TFDs, denoted as M_z^{S}, and calculated as:

$$M_z^{\mathrm{S}} = \frac{1}{N_t N_f}\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\rho_{z_N}(t,f)|^{\frac{1}{p_{\mathrm{S}}}}\, dtft\right]^{p_{\mathrm{S}}}, \qquad p_{\mathrm{S}} > 1. \tag{2.35}$$

Normalization of the TFDs can be performed in various ways, leading to a different definitions of the concentration measure. In this thesis, the TFD normalization has been performed w.r.t. its total energy, resulting in $E_{z_N} = 1$:

$$\rho_{z_N}(t,f) = \frac{\rho_z(t,f)}{E_z} = \frac{\rho_z(t,f)}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \rho_z(t,f)dtdf}. \tag{2.36}$$

The smaller the M_z^{S} value is, the higher TFD concentration is achieved, and vice versa. Furthermore, the measure proposed by Stanković does not suffer from the same problem as the previously discussed Jones-Park concentration measure, in case when the signal components have different amplitudes. In order to achieve the robustness of the concentration measure, it has been recommended in [68] to use a low order form (e.g. with $p_{\mathrm{S}} = 2$), since in general, there are no sharp edges between the zero and non-zero TFD values, hence the proposed measure would be more sensitive to smaller TFD values when a higher order form is used.

Alternative way to calculate a measure of the TFD complexity is with the Rényi entropy, denoted as $R_z^{\alpha_{\mathrm{R}}}$, and calculated as [81, 6]:

$$R_z^{\alpha_{\mathrm{R}}} = \frac{1}{1-\alpha_{\mathrm{R}}}\log_2\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \rho_{z_N}^{\alpha_{\mathrm{R}}}(t,f)dtdf\right], \qquad \alpha_{\mathrm{R}} > 2, \tag{2.37}$$

where the parameter α_{R} is order of the Rényi entropy. The Rényi entropy has a larger values for the less concentrated TFDs, and vise versa. It has been shown in [6] when choosing α_{R} to be an odd integer, the oscillatory cross-terms are cancelled under integration, thus the Rényi entropy counts only information originating from the auto-terms. In TFD applications, the 3rd order Rényi entropy is commonly used [81, 6].

Example 2.7. In this example, the concentration measures given by the (2.32), (2.35), and (2.37) with $p_{\mathrm{S}} = 2$ and $\alpha_{\mathrm{R}} = 3$ have been calculated for the selected QTFDs of signals $z_{\mathrm{LFM}}(t)$, $z_{\mathrm{nonLFM}}(t)$, and $z_{\mathrm{bat}}(t)$ previously defined in Examples 2.4 - 2.6. The results are presented in Table 2.3, along with the concentration measures of the ideal signal TFDs

Table 2.3: Concentration measures M_z^{JP}, $R_z^{\alpha_{\mathrm{R}}}$, and M_z^{S} of the three component LFM signal, $z_{\mathrm{LFM}}(t)$, the two component, $z_{\mathrm{nonLFM}}(t)$, and the bat echolocation signal, $z_{\mathrm{bat}}(t)$. The bold value indicates the best performing TFD according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	M_z^{JP} $\cdot 10^4$	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{s}}=2$	M_z^{JP} $\cdot 10^4$	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{s}}=2$	M_z^{JP} $\cdot 10^4$	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{s}}=2$
Ideal	19.6078	9.2568	0.0111	19.6078	8.9944	0.0078	–	–	–
WVD	5.3164	**9.8680**	3.1258	2.4245	**10.3746**	4.6003	3.8479	**10.8951**	2.0656
Levin	0.6766	12.0980	4.2220	0.9008	12.0779	3.9331	0.6549	12.4221	2.5702
Born-Jordan	2.4248	11.8764	1.4278	2.1967	11.9253	1.6491	2.5431	11.9996	0.9127
Sinc	2.7659	11.3981	2.0831	2.1684	11.5066	2.3329	2.8403	11.5626	1.5561
Rihaczek	0.6615	12.0711	4.4520	0.8468	12.0333	4.3354	0.6526	12.4315	2.6449
Page	0.8698	11.7608	4.9469	1.3271	11.4657	4.9866	0.8441	11.9832	3.1353
Choi-Williams	3.0344	11.7513	1.1793	2.5217	11.7957	1.4661	3.4363	11.6805	0.8181
B	**7.6235**	11.2217	0.7890	**4.7245**	11.8499	1.0676	**4.5044**	11.4664	0.6511
Modified B	7.5494	11.1341	0.8793	4.6801	11.8070	1.1135	4.4024	11.4231	0.6868
Extended modified B	2.2874	13.1718	**0.5283**	2.5938	13.4071	**0.5168**	2.3139	13.2663	**0.2893**
RGK-TFD	22.3985	10.3326	0.3203	23.1350	10.7851	0.7507	14.5322	9.7164	0.5270

(highlighted row), which are the best obtainable measure values for the given signal.

Since the Rényi entropy does not take into account the cross-terms, the best performing TFD is selected as the one with the most concentrated auto-terms. As discussed in Section 2.3.4, the WVD has the best concentration, and other TFDs are derived through a low-pass filtering, resulting in the lower auto-terms concentration, but better cross-term suppression, hence, the Rényi entropy will always favour the WVD as the best performing TFD. As it can be seen from the obtained results, the remaining two concentration measures select a different kernel as the optimal one; however, by visual inspection of the calculated QTFDs, shown in Figs. 2.8 - 2.10, it can be concluded that the concentration measure proposed by Stanković achieves the best result among the considered concentration measures.

Because of the issues described in Example 2.7, the problem of objective TFD performance assessment is an ongoing topic of research. A more detailed measure which takes into account the auto-terms and the cross-terms geometry is presented in [17]. The best performing TFD can be found by solving an optimization problem with the selected TFD performance measure being an objective function which needs to be minimized or maximized, depending on the selected measure. An example of such optimization is the adaptive STFT with the Gaussian window calculated as [40]:

$$w_{t,f}(\tau) = (-2\,\mathrm{Re}\{c_{t,f}\}/\pi)^{1/4} \exp[c_{t,f}(\tau - t^2)] e^{-j2\pi f\tau}, \tag{2.38}$$

where $c_{t,f}$ is parameter calculated by selecting (2.34) as an objective function of the optimization problem.

2.4.2 Optimal Radially Gaussian Kernel

Baraniuk and Jones [5] developed a method for finding an optimal kernel, $g_{\text{opt}}(\nu,\tau)$, by solving the optimization problem with the objective function defined as:

$$g_{\text{opt}}(\nu,\tau) = \arg\max_{g(\nu,\tau)} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |A_z(\nu,\tau)g(\nu,\tau)|^2 d\tau d\nu, \tag{2.39}$$

with the following constraints ensuring that the kernel has properties of the low-pass filter:

$$g(0,0) = 1, \tag{2.40a}$$

$$g(r_1,\psi) \geq g(r_2,\psi), \qquad \forall r_1 < r_2, \qquad \forall \psi, \tag{2.40b}$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |g(\nu,\tau)|^2 d\tau d\nu \leq \alpha_{\text{RGK}}, \qquad \alpha_{\text{RGK}} \geq 0, \tag{2.40c}$$

where (r_i,ψ) are the AF polar coordinates, while the parameter α_{RGK} controls the volume under the kernel, that is, a trade-off between the auto-terms concentration and the cross-terms suppression (with $\alpha_{\text{RGK}} \to \infty$, the kernel becomes the WVD kernel, i.e. $g(\nu,\tau) = 1$).

Maximizing the objective function (2.39) will ensure that signal auto-terms are located in the $g_{\text{opt}}(\nu,\tau)$ pass-band, without their energy being attenuated. The reason behind this is the fact that the cross-terms total energy is zero, that is:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} W_{z_{i,j}}(t,f)dtdf = 0, \tag{2.41}$$

thus all of the signal energy is concentrated in the auto-terms. The solution of the optimization problem gives $1/0$ values, and this is why the kernel obtained by (2.39) is referred as the $1/0$ kernel. However, the downside of the $1/0$ kernel are sharp edges between the kernel pass-band and stop-band, which may cause a Gibbs phenomenon (i.e. ringing artifacts). This effect can be negated by forcing the kernel to be smooth with a Gaussian function [4]:

$$g(\nu,\tau) = e^{-(\nu^2+\tau^2)/2\sigma^2(\psi)}, \tag{2.42}$$

where $\sigma(\psi)$ is the spread function, which controls the kernel spread along the radial angle ψ. The kernel obtained by the (2.42) is referred as the Radially Gaussian Kernel (RGK)[2].

Example 2.8. In this example, the RGKs, and the associated TFDs of the previously defined signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$ have been calculated. The results are shown

[2] Available at: `http://www.ece.rice.edu/dsp/software/TFA/RGK/rgk.tar.Z`

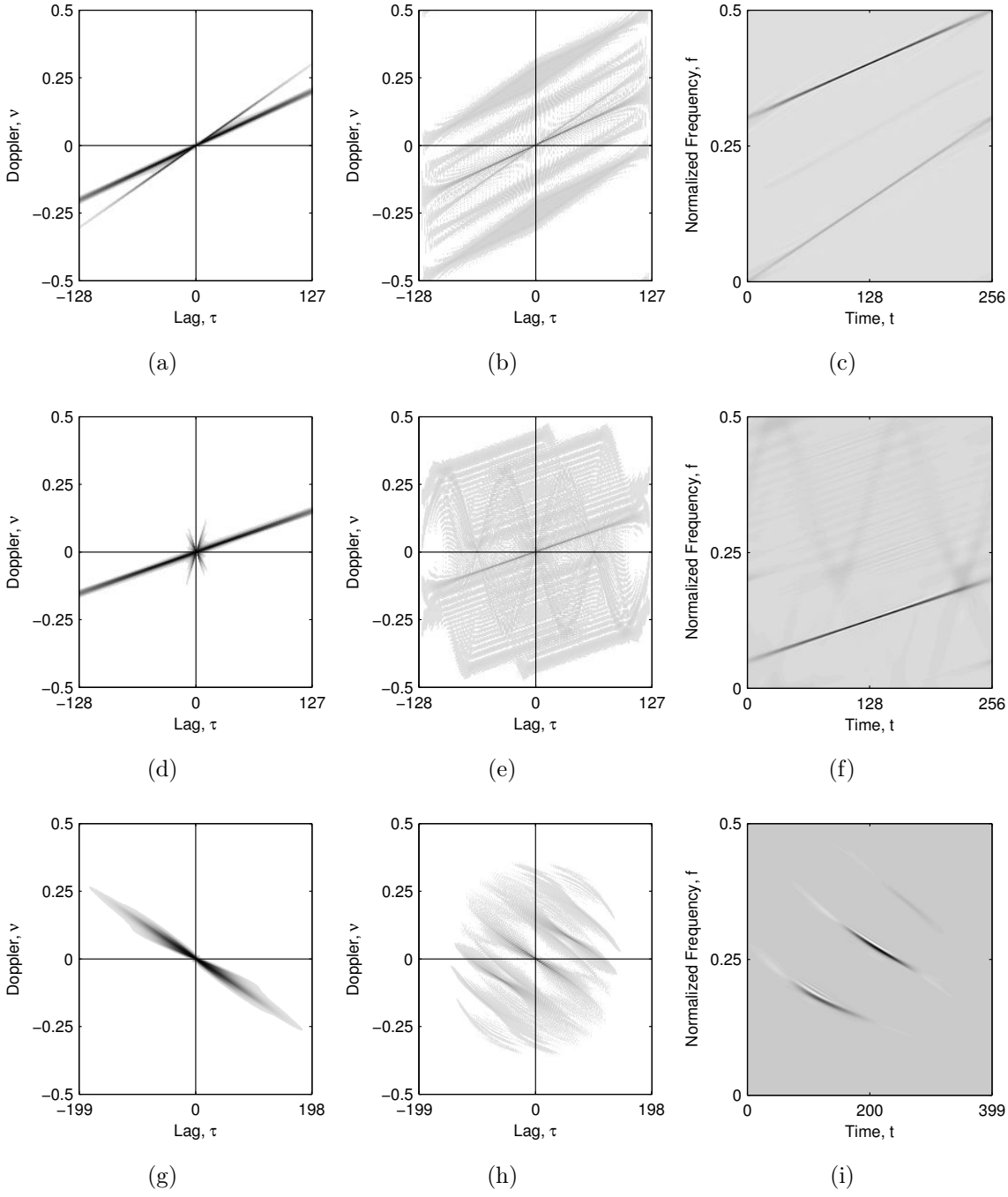


Figure 2.11: Performance of the optimal radially Gaussian kernel: (a)-(c) the RGK, the non filtered AF, and the associated TFD ($M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 0.3203$, $R_z^{\alpha_{\mathrm{R}}}|_{\alpha_{\mathrm{R}}=3} = 10.3326$) of the three LFM component signal, $z_{\mathrm{LFM}}(t)$, respectively; (d)-(f) the RGK, the non filtered AF, and the associated TFD ($M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 0.7507$, $R_z^{\alpha_{\mathrm{R}}}|_{\alpha_{\mathrm{R}}=3} = 10.7851$) of the two component signal, $z_{\mathrm{nonLFM}}(t)$, respectively; (g)-(i) the RGK, the non filtered AF, and the associated TFD ($M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 0.5270$, $R_z^{\alpha_{\mathrm{R}}}|_{\alpha_{\mathrm{R}}=3} = 9.7164$) of the bat echolocation signal, $z_{\mathrm{bat}}(t)$, respectively.

in Fig. 2.11, with the kernel volume parameter selected as $\alpha_{\text{RGK}} = 5$, while the concentration measures of the resulting TFDs have been shown in Table 2.3. When compared to the non-adaptive TFDs calculated in Examples 2.4 - 2.6 the performance of the RGK is significantly better; however, it is not perfect. For the three LFM component signal, $z_{\text{LFM}}(t)$, the RGK followed the auto-terms of the two strongest components, but missed to capture the weakest component, as shown in Fig. 2.11(c). On the other hand, in case of the two component signal, $z_{\text{nonLFM}}(t)$, the RGK captured a LFM component, however it missed the sinusoidal frequency modulated component, as shown in Fig. 2.11(f).

2.5 Summary

In this chapter, advantages of the joint time-frequency signal representation over the one-dimensional time and frequency signal representation have been shown on the synthetical ($z_{\text{LFM}}(t)$ and $z_{\text{nonLFM}}(t)$) and real-life ($z_{\text{bat}}(t)$) signals. Properties of the ideal TFD have been reviewed, and the classical numerical methods for the TFD calculation have been introduced. The more advanced adaptive TFD calculation methods have been discussed, and necessity of the objective TFD performance assessment and the TFD optimization has been addressed.

Chapter 3

Compressive Sensing and Sparse Filtering

In this chapter, the advantages and disadvantages of the CS approach over the classical sampling approach dictated by the Shannon-Nyquist sampling theorem are discussed. Short overview of the CS related research is given, and application for the TFD concentration enhancement is thoroughly described. The key conditions which have to be satisfied in order to guaranty the meaningful solution of the sparse signal reconstruction and their TFD related implications are discussed.

Furthermore, the sparse signal reconstruction based on the ℓ_0 norm, the ℓ_1 norm, and the ℓ_2 norm minimization is described in details, and associated state-of-the-art sparse reconstruction algorithms are introduced. The state-of-the-art sparse signal reconstruction algorithms are modified and tested for the reconstruction of the sparse TFDs of synthetical and real-life signals.

3.1 Introduction

3.1.1 Motivation

The continuous-time and the discrete-time signals are connected through a process called sampling, which implies a periodical selection of samples from the continuous-time signal with a sampling period T_s, that is:

$$s_s(t) = s(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s), \tag{3.1}$$

where $s_s(t)$ is the sampled continuous-time signal. The Fourier transformation of this signal can be calculated as:

$$\begin{aligned} S_s(f) &= \mathcal{F}\left\{ s(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \right\} = S(f) * \mathcal{F}\left\{ \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \right\} \\ &= S(f) * \frac{1}{T_s} \sum_{n=-\infty}^{\infty} \delta(f - nf_s) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} S(f - nf_s), \end{aligned} \tag{3.2}$$

which states that the spectrum of the discrete-time signal is composed of infinite copies of the continuous-time signal spectrum repeating itself every sampling frequency $f_s = 1/T_s$. If a signal of bandwidth B, is sampled with a sampling frequency $f_s \geq 2B$, the original continuous-time signal can be easily reconstructed by singling-out the spectrum copy for $n = 0$, by means of the low-pass filter with a cut-off frequency $f_s/2$. However, in case when a signal is sampled with a sampling frequency $f_s < 2B$, the signal reconstruction is not possible, since the spectrum copies overlap with each other, making extraction of a single spectrum copy impossible. This effect is called the aliasing, and can be avoided by a proper selection of the sampling frequency as $f_s \geq 2B$. The criterium $f_s \geq 2B$ is well known as the Shannon-Nyquist sampling theorem, while the frequency $f_N = 2B$ is called the Nyquist frequency.

In some special cases, when the signal behaviour is *a priori* known, the Shannon-Nyquist sampling theorem can be circumvented, allowing sampling with the sub-Nyquist frequencies. Recently developed sampling methods exploit the signal sparsity, which means that the signal can be represented in a certain domain with K non-zero coefficients, where $K << N_t$. For example, a sinusoidal signal can be represented with only one sample in the frequency domain. Most signals are non-sparse in the domain of interest, but can become sparse (or approximately sparse) by applying a domain transformation. The sparse domain is then used as the *a priori* knowledge, allowing signal undersampling, also called the compressive sensing. Traditionally, CS implies signal undersampling with samples randomly picked [27, 3, 20, 48]; however, the samples can be picked to favour specific signal features, while discarding the others [69, 64, 65].

Compressive sensing has a number of advantages, primarily the sampling hardware requirements can be eased considerably and the CS signals require less storage space. However, the CS benefits are gained in expense of the sparse signal reconstruction algorithm mathematical complexity. Reconstruction of the traditionally sampled signal, as already mentioned, involves only a low-pass filter, while on the other hand, the CS signal reconstruction involves a specially designed iterative algorithms for solving a unconstrained linear optimization problem.

3.1.2 Compressive Sensing Development and Applications

Presumably, the first contribution to the sparse signal recovery has been reported in [57] as far back as 1795 by the French mathematician de Prony. The method decomposed a nonharmonic trigonometric sum: $y(t) = \sum_{j=1}^{N} x_j e^{2\pi i f_j t}$ into its basic components by finding the frequencies $f_j \in \mathbb{R}$ and the associated amplitudes $x_j \in \mathbb{C}$. In modern times, the empirical beginnings of the sparse signal recovery can be traced back to the late 1970's, when the sparse signal reconstruction algorithm has been successfully employed in the reflection seismology, in order to calculate a sparse reflection between the subsurface layers [23, 73]. At the same time, a similar method has been applied in radio astronomy for the aperture synthesis [37]. In the late 1980's, a method for the sparse signal recovery has been used in the spectroscopy [63] and in the nuclear magnetic resonance (NMR) spectroscopy [9] in order to recover a spectrum of the undersampled signal. In image processing, the sparse signal recovery methods have been used in order to remove noise from the heavily noise corrupted images, dating back to the early 1990's [62]. The sparsity based method has also been used in the statistics as the least absolute shrinkage and selection operator (LASSO) [74].

The common idea in all of the above discussed applications is to a solve system of under-determined equations with the ℓ_1 norm minimization in the sparse domain. However, the reason behind the method effectiveness at that time were unknown. The major breakthrough in the field of sparse signal recovery has been reported in [21], by giving a mathematical proof behind the method effectiveness and introducing the restricted isometry property (RIP) (initially called the uniform uncertainty principle). In the same paper, it has been shown that the Fourier matrices satisfy this property, opening a wide application in the field of signal processing. The initial paper has been followed by [27, 26], in which the term compressive sensing has been formulated.

The CS has already had a significant impact in the various fields, although the major breakthrough point of the CS related research happened not so long ago and it is still an ongoing topic of research. Some of the more impactful CS applications are listed in the sequel of this section.

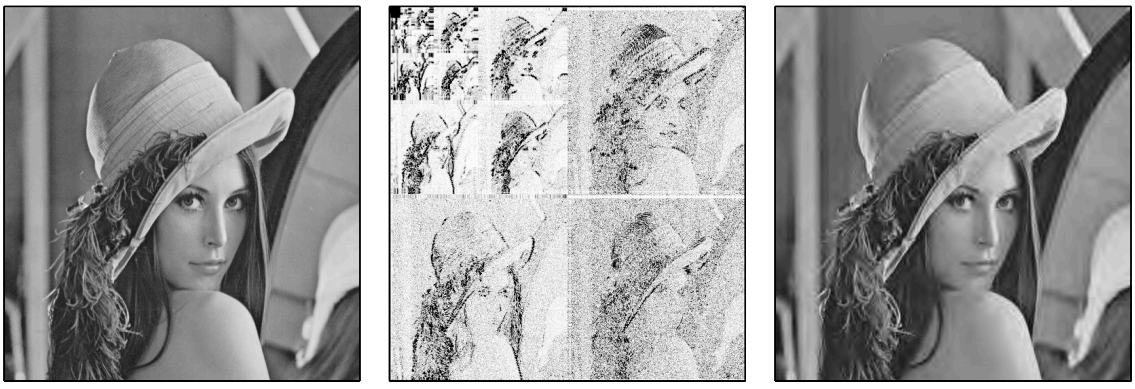
- **Sparse approximation.** Sparse signal approximation methods for various signal types have been developed before the introduction of CS, and they are the fundamental basis in most of the modern compression methods. The idea behind the sparse approximation is to find a certain domain in which the observed signal is sparse (or approximately sparse) and store only a small number of the highest value samples in the sparse domain. Original signal can be then approximated by returning the stored samples to the original observation domain. Most of the modern compression techniques (e.g. JPEG, MPEG, MP3, etc.) work in this way, for example a wavelet transformation provides a good sparse representation of images, while a Gabor ex-

pansion is suitable for the audio signals. Example of image approximation using the wavelet coefficients is shown in Fig. 3.1.

- **Signal acquisition.** The previously discussed sparse approximation method describes a typical up-to-date acquisition and storage system. First, the full set of signal samples is measured, and then most of the samples are discarded in the sparse domain through the signal compression. The CS theory provides an alternative approach to such a wasteful acquisition system. Instead of acquiring a full set of the signal samples (which is a strain on the acquisition hardware), only a small subset of the signal samples is measured, enough to reconstruct the most significant samples in the sparse domain. Example of such system is the single-pixel camera [28], and the CMOS image sensor with the programmable CS [53].

- **Medical imaging.** In the similar way, the CS has been successfully implemented in the magnetic resonance imaging (MRI) [48, 75], and the computed tomography (CT) [7], allowing the faster signal acquisition (some MRI scans can take up to few hours), and patients are exposed to a lower amount of radiation (the main problem in the CT acquisition is a trade-off between the CT quality and the radiation exposure). In a similar way, the CS is implemented in the related problems of spectroscopy [63] and NMR spectroscopy [9, 70].

In the field of time-frequency signal processing, the CS based methods have been used in radar systems in order to enhance their resolution [56, 66]. The similar problem is formulated in sonar [45] and in wireless communication networks [38]. Furthermore, the CS has also been used in astrophysics for the gravitational wave data analysis [1], in geophysics for the hydrocarbon detection [33], in medical imaging for the high resolution EEG [35], etc. Application of CS for the TFD concentration enhancement, explored in



(a) (b) (c)

Figure 3.1: Image compression using the wavelet coefficients: (a) original uncompressed image, (b) wavelet coefficients (Symlets 8, decomposition level 5), (c) obtained image by taking only a 1% of the highest value wavelet coefficients (99% of the wavelet coefficients are set to zero).

the [30, 69, 54, 78, 31, 77] is topic of this thesis, and it is more thoroughly discussed in the continuation of this chapter.

3.2 Compressive Sensing for TFD Concentration Enhancement

3.2.1 Compressed Sensed Ambiguity Function

Ideal TFDs are inherently sparse (as shown in Fig. 2.1(b)), since they are composed from the components IFs, thus requiring only one sample per component and frequency bin, that is: $N_c \cdot N_t << N_f \cdot N_t$ samples in total, hence the CS and the sparse filtering can be utilized. The CS-AF, $\mathbf{A}'_z(\nu,\tau)$, also known as the observation or measurement matrix, is formulated as:

$$\mathbf{A}'_z(\nu,\tau) = \boldsymbol{\phi}(\nu,\tau) \odot \mathbf{A}_z(\nu,\tau), \tag{3.3}$$

where the operator $\odot$ denotes a element-by-element matrix multiplication, $\mathbf{A}_z(\nu,\tau)$ is the matrix representation of the AF, and $\boldsymbol{\phi}(\nu,\tau)$ is the sensing matrix. Unlike the previously discussed CS applications in which the sensing matrix randomly picks a small subset of the original data, in the TFD related applications the CS area is chosen in such a way which does not contain the cross-terms. Thus, the sensing matrix defines the $N'_\nu \times N'_\tau$ area Ω around the AF plane origin:

$$\boldsymbol{\phi}(\nu,\tau) = \begin{cases} 1, & (\nu,\tau) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

The sensing matrix defined in this way discards the highly oscillatory cross-terms located away from the AF domain origin, while preserving the auto-terms located closer to the AF origin. Proper selection of N'_τ and N'_ν is crucial for TFD localization and cross-terms suppression. Too few samples will reduce time-frequency localization, which results in a blurred TFD. With too many samples, on the other hand, the CS-AF area will contain the cross-terms along with the auto-terms, resulting in a cross-term corrupted TFD. Most authors use a square CS-AF area with $N'_\tau = N'_\nu \approx \sqrt{N_t}$ samples [64, 18, 30], which gives: $\text{card}(\mathbf{A}'_z(\nu,\tau)) \approx N_t$, where the operator $\text{card}(\mathbf{x})$ denotes the cardinality of $\mathbf{x}$, that is a number of non-zero elements in $\mathbf{x}$.

In the standard CS notation, matrix multiplication is commonly used to define a connection between the observation and solution matrices, thus from (2.27) and (3.3):

$$\mathbf{A}'_z(\nu,\tau) = \mathbf{W}(\nu,t) \cdot \boldsymbol{\vartheta}_z(t,f) \cdot \mathbf{W}_i^H(\tau,f) = \boldsymbol{\psi} \cdot \boldsymbol{\vartheta}_z(t,f), \tag{3.5}$$

where $\boldsymbol{\vartheta}_z(t,f)$ is a matrix representation of the sparse TFD reconstructed from the CS-AF, the operator $\cdot^H$ denotes the Hermitian transpose, while $\mathbf{W}(\nu,t)$ and $\mathbf{W}_i(\tau,f)$ are matrices representing a Fourier transformation and its inverse, respectively, defined as:

$$\mathbf{W}(\nu,t)=\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & \left(e^{-2\pi i/N_\nu}\right)^1 & \left(e^{-2\pi i/N_\nu}\right)^2 & \cdots & \left(e^{-2\pi i/N_\nu}\right)^{N_t-1}\\ 1 & \left(e^{-2\pi i/N_\nu}\right)^2 & \left(e^{-2\pi i/N_\nu}\right)^4 & \cdots & \left(e^{-2\pi i/N_\nu}\right)^{2(N_t-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \left(e^{-2\pi i/N_\nu}\right)^{N_\nu-1} & \left(e^{-2\pi i/N_\nu}\right)^{2(N_\nu-1)} & \cdots & \left(e^{-2\pi i/N_\nu}\right)^{(N_\nu-1)(N_t-1)}\end{bmatrix}, \quad (3.6a)$$

$$\mathbf{W}_i(\tau,f)=\frac{1}{N_\tau}\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & \left(e^{2\pi i/N_\tau}\right)^1 & \left(e^{2\pi i/N_\tau}\right)^2 & \cdots & \left(e^{2\pi i/N_\tau}\right)^{N_f-1}\\ 1 & \left(e^{2\pi i/N_\tau}\right)^2 & \left(e^{2\pi i/N_\tau}\right)^4 & \cdots & \left(e^{2\pi i/N_\tau}\right)^{2(N_f-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \left(e^{2\pi i/N_\tau}\right)^{N_\tau-1} & \left(e^{2\pi i/N_\tau}\right)^{2(N_\tau-1)} & \cdots & \left(e^{2\pi i/N_\tau}\right)^{(N_\tau-1)(N_f-1)}\end{bmatrix}, \quad (3.6b)$$

and $\boldsymbol{\psi}$ is a domain transformation matrix, representing a two-dimensional Fourier transformation $\boldsymbol{\psi}=\mathbf{W}(\nu,t)\otimes\mathbf{W}_i(\tau,f)$, where the operator $\otimes$ denotes the Kronecker product, defined as:

$$\boldsymbol{\psi}=\mathbf{W}(\nu,t)\otimes\mathbf{W}_i(\tau,f)=\begin{bmatrix}w_{11}\mathbf{W}_i(\tau,f) & w_{12}\mathbf{W}_i(\tau,f) & \cdots & w_{1N_t}\mathbf{W}_i(\tau,f)\\ w_{21}\mathbf{W}_i(\tau,f) & w_{22}\mathbf{W}_i(\tau,f) & \cdots & w_{2N_t}\mathbf{W}_i(\tau,f)\\ \vdots & \vdots & \ddots & \vdots\\ w_{N_\nu 1}\mathbf{W}_i(\tau,f) & w_{N_\nu 2}\mathbf{W}_i(\tau,f) & \cdots & w_{N_\nu N_t}\mathbf{W}_i(\tau,f)\end{bmatrix}$$

$$=\begin{bmatrix}w_{11}w_{i11} & w_{11}w_{i12} & \cdots & w_{11}w_{i1N_f} & \cdots & \cdots & w_{1N_t}w_{i11} & w_{1N_t}w_{i12} & \cdots & w_{1N_t}w_{i1N_f}\\ w_{11}w_{i21} & w_{11}w_{i22} & \cdots & w_{11}w_{i2N_f} & \cdots & \cdots & w_{1N_t}w_{i21} & w_{1N_t}w_{i12} & \cdots & w_{1N_t}w_{i2N_f}\\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots\\ w_{11}w_{iN_\tau 1} & w_{11}w_{iN_\tau 2} & \cdots & w_{11}w_{iN_\tau N_f} & \cdots & \cdots & w_{1N_t}w_{iN_\tau 1} & w_{1N_t}w_{iN_\tau 2} & \cdots & w_{1N_t}w_{iN_\tau N_f}\\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots\\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots\\ w_{N_\nu 1}w_{i11} & w_{N_\nu 1}w_{i12} & \cdots & w_{N_\nu 1}w_{i1N_f} & \cdots & \cdots & w_{N_\nu N_t}w_{i11} & w_{N_\nu N_t}w_{i12} & \cdots & w_{N_\nu N_t}w_{i1N_f}\\ w_{N_\nu 1}w_{i21} & w_{N_\nu 1}w_{i22} & \cdots & w_{N_\nu 1}w_{i2N_f} & \cdots & \cdots & w_{N_\nu N_t}w_{i21} & w_{N_\nu N_t}w_{i22} & \cdots & w_{N_\nu N_t}w_{i2N_f}\\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots\\ w_{N_\nu 1}w_{iN_\tau 1} & w_{N_\nu 1}w_{iN_\tau 2} & \cdots & w_{N_\nu 1}w_{iN_\tau N_f} & \cdots & \cdots & w_{N_\nu N_t}w_{iN_\tau 1} & w_{N_\nu N_t n}w_{iN_\tau 2} & \cdots & w_{N_\nu N_t n}w_{iN_\tau N_f}\end{bmatrix}. \quad (3.7)$$

Kronecker product usually results in a very large matrices, as it can be seen from (3.7). The two-dimensional Fourier transformation matrix, $\boldsymbol{\psi}=\mathbf{W}(\nu,t)\otimes\mathbf{W}_i(\tau,f)$, is a matrix containing $N_\nu N_t \times N_\tau N_f$ elements, which if implemented in such a way would require significant computational power and storage space. This is why, in most practical realizations, $\boldsymbol{\psi}$ and its inverse, $\boldsymbol{\psi}^H$ are implemented as functions, resulting in a significantly

faster algorithms [29, 2]. In this thesis multiplication with $\boldsymbol{\psi}$ and its inverse $\boldsymbol{\psi}^H$ are implemented as a MATLAB functions, `fun_ambi2tf.m` and `fun_tf2ambi.m`, the codes of which are given in Appendix A.

Since $\mathbf{A}'_z(\nu,\tau)$ has cardinality of $\text{card}(\mathbf{A}'_z(\nu,\tau)) = N'_\nu \cdot N'_\tau$ samples, and $\boldsymbol{\vartheta}_z(t,f)$ has cardinality of $\text{card}(\boldsymbol{\vartheta}_z(t,f)) = N_t \cdot N_f$ samples, where $N'_\nu \cdot N'_\tau << N_t \cdot N_f$, the system (3.5) is under-determined, thus $\boldsymbol{\vartheta}_z(t,f)$ can have an infinite number of possible solutions and the goal of the sparse reconstruction algorithm is to find the optimal solution to:

$$\boldsymbol{\vartheta}_z(t,f) = \boldsymbol{\psi}^H \cdot \mathbf{A}'_z(\nu,\tau) = \boldsymbol{\psi}^H \cdot [\boldsymbol{\phi}(\nu,\tau) \odot \mathbf{A}_z(\nu,\tau)]\,. \tag{3.8}$$

3.2.2 Restricted Isometry and Mutual Incoherence Conditions

In this section two main conditions for a successful reconstruction of the sparse signals are discussed: the restricted isometry property, and the mutual incoherence between the sensing and domain transforming matrices. The mathematical background behind these two conditions has been given in [21, 27], while discussion in this section is focused on their implications in the standard and TFD related applications.

In the standard CS formulation, which implies a random selection of samples, the number of selected samples is dictated by the RIP condition [21]. By enforcing the RIP condition, the CS-AF area size should contain $\text{card}(\mathbf{A}'_z(\nu,\tau)) \approx N_c N_t \log(N_t N_f)$ samples, with the lower limit of $\text{card}(\mathbf{A}'_z(\nu,\tau)) \approx N_c N_t \log(N_f/N_c)$ samples [21, 20, 61, 30]. However, in the TFD related applications the goal is not to exactly reconstruct a starting TFD, since this would result in the cross-term corrupted WVD. Instead, the goal is to obtain a new WVD-like TFD, but with highly suppressed cross-terms and localized auto-terms. This is a reason why the RIP condition in the TFD related applications is not enforced, and the CS-AF area used in literature, containing approximately $\text{card}(\mathbf{A}'_z(\nu,\tau)) \approx \sqrt{N_t} \times \sqrt{N_t} = N_t$ samples, is of the order lower than the number of samples dictated by the RIP condition.

Furthermore, in the standard CS approaches it is imperative that the matrices $\boldsymbol{\psi}$ and $\boldsymbol{\phi}(\nu,\tau)$ have a low mutual coherence in order to guaranty a meaningful solution of the sparse reconstruction [27, 20]. The mutual coherence is calculated as the largest cross-correlation between any two elements of matrices $\boldsymbol{\psi}$ and $\boldsymbol{\phi}(\nu,\tau)$, that is:

$$\mu\left(\boldsymbol{\phi}(\nu,\tau), \boldsymbol{\psi}\right) = \max_{i \neq j} \frac{|\langle \boldsymbol{\phi}_i(\nu,\tau), \boldsymbol{\psi}_j \rangle|}{||\boldsymbol{\phi}_i(\nu,\tau)||^2 \, ||\boldsymbol{\psi}_j||^2}. \tag{3.9}$$

With regards to matrix: $\boldsymbol{\Upsilon} = \boldsymbol{\psi} \cdot \boldsymbol{\phi}(\nu,\tau)$, the mutual coherence is calculated as the largest element of the normalized Gramian matrix $\boldsymbol{\Upsilon}^H \boldsymbol{\Upsilon}$ [26], that is:

$$\mu(\boldsymbol{\Upsilon}) = \max_{i \neq j} \frac{|\langle \boldsymbol{\Upsilon}_i, \boldsymbol{\Upsilon}_j \rangle|}{||\boldsymbol{\Upsilon}_i||^2 \, ||\boldsymbol{\Upsilon}_j||^2}. \tag{3.10}$$

The reason behind a low mutual coherence requirement between the matrices $\boldsymbol{\psi}$ and $\boldsymbol{\phi}(\nu,\tau)$ is to ensure that the signal is not sparse in the sensing domain, that is, to ensure that both $\boldsymbol{\vartheta}_z(t,f)$ and $\mathbf{A}_z(\nu,\tau)$ are not sparse. This condition is of vital importance for the CS, because sampling in a sparse domain will very likely result that most of the selected samples are equal to zero, providing very little information about the signal energy. Instead, sampling has to be performed in a such sensing domain, which when transformed back to the sparse domain, will spread throughout the entire sparse domain, providing as much information about the signal energy as possible.

By observing both TF and AF domain, it can be concluded that both TF and AF domain are sparse; they are both composed from the auto-terms and the cross-terms between them, while most of the samples can be approximated to zero. The difference between the domains is mainly in their geometries, that is, the cross-term and the auto-term location, as shown in Fig. 2.5. This means that the TF and AF domain are mutually coherent, and a random selection of samples in the AF domain most likely will not produce a correct sparse solution in the TF domain. However, as discussed in Section 3.2.1, compressive sensing of the AF is not performed randomly; samples are selected in such a way which ensures inclusion of the auto-terms and exclusion of the cross-terms, making sure that the CS-AF area contains the useful information. This is why the low mutual coherence condition can also be circumvented along with the RIP condition in the TFD related applications.

3.3 Sparse TFD Reconstruction

3.3.1 Problem Formulation

Problem described in (3.8) is a well-known unconstrained linear optimization problem [25, 12, 29, 82, 10, 2], and thus it can be rewritten as:

$$\widehat{\boldsymbol{\vartheta}}_z(t,f) = \arg\min_{\boldsymbol{\vartheta}_z(t,f)} \frac{1}{2}\left|\left|\boldsymbol{\vartheta}_z(t,f) - \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau)\right|\right|_2^2 + \lambda c\left(\boldsymbol{\vartheta}_z(t,f)\right), \tag{3.11}$$

where regularization function $c\left(\boldsymbol{\vartheta}_z(t,f)\right) : \mathbb{R}^2 \to \mathbb{R}$ is a nonsmooth and nonconvex, λ is a regularization parameter, and $f\left(\boldsymbol{\vartheta}_z(t,f)\right) = \frac{1}{2}\left|\left|\boldsymbol{\vartheta}_z(t,f) - \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau)\right|\right|_2^2$ is an error estimating function.

Since the problem of estimating $\widehat{\boldsymbol{\vartheta}}_z(t,f)$ is ill-posed, a regularization function has to be applied as some sort of prior information about the signal, in order to find a proper solution. Regularization functions are usually sparsity inducing such as ℓ_p norm, and in most problems the ℓ_1 norm based regularization function is commonly used [18, 30, 33, 64, 65, 66, 78, 31, 77]. Optimization problem (3.11) can be solved alternatively, with the greedy algorithms, such as matching pursuit and orthogonal matching pursuit

[50, 52, 19, 84, 54], which are not based on the explicit optimization formulation; however, due to extensiveness of the area, the discussion in this thesis has been limited to the methods based on the ℓ_0 norm, the ℓ_1 norm, and the ℓ_2 norm minimization.

3.3.2 ℓ_2 Norm Based TFD Reconstruction

The ℓ_2 norm based regularization function, $c^{\ell_2}(\boldsymbol{\vartheta}_z(t,f))$, has the same form as the error estimating function, that is:

$$c^{\ell_2}(\boldsymbol{\vartheta}_z(t,f)) = ||\boldsymbol{\vartheta}_z(t,f)||_2^2 = \sum^{N_t}\sum^{N_f} |\boldsymbol{\vartheta}_z(t,f)|^2 , \tag{3.12}$$

thus the optimization problem (3.11) becomes a problem of $\boldsymbol{\vartheta}_z(t,f)$ energy minimization, that is:

$$\begin{aligned} \boldsymbol{\vartheta}_z^{\ell_2}(t,f) &= \arg\min_{\boldsymbol{\vartheta}_z(t,f)} ||\boldsymbol{\vartheta}_z(t,f)||_2^2 , \\ &\text{subject to: } \left|\left|\boldsymbol{\vartheta}_z(t,f) - \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau)\right|\right|_2^2 \leq \epsilon, \end{aligned} \tag{3.13}$$

where ϵ is a user defined energy threshold value which controls the reconstruction accuracy. Since the ℓ_2 norm is used in both regularization and error estimating function, the solution of the optimization problem (3.13) is trivial:

$$\boldsymbol{\vartheta}_z^{\ell_2}(t,f) = \boldsymbol{\psi}^H \left(\boldsymbol{\psi}\boldsymbol{\psi}^H\right)^{-1} \mathbf{A}'_z(\nu,\tau) = \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau). \tag{3.14}$$

Example 3.1. In this example, the reconstructed sparse TFDs based on the ℓ_2 norm minimization have been calculated for the signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$. In order to ensure that N'_τ and N'_ν are odd, the CS-AF area has been calculated as $N'_\tau = N'_\nu = 2\lceil\sqrt{N_t}/2\rceil - 1$, resulting in $N'_\tau = N'_\nu = 15$ samples for the signals $z_{\text{LFM}}(t)$ and $z_{\text{nonLFM}}(t)$, and in $N'_\tau = N'_\nu = 19$ samples for the signal $z_{\text{bat}}(t)$, as shown in Figs. 3.2(a) - 3.2(c). The resulting ℓ_2 norm based reconstructed sparse TFDs are shown in Figs. 3.2(d) - 3.2(f). By comparing the obtained concentration measures with the ones previously obtained for the classically filtered TFDs, listed in Table 2.3, it can be seen that the obtained M_z^{S} values are significantly larger, while the obtained Rényi entropy values are even lower then the Rényi entropy of the ideal TFD. This means that the concentration of the auto-terms is very poor and a significant portion of the auto-terms got filtered out as well, which can be confirmed by the visual inspection of Figs. 3.2(d) - 3.2(f).

As it can be seen from (3.14) and confirmed in Example 3.1, the solution based on the ℓ_2 norm minimization, $\boldsymbol{\vartheta}_z^{\ell_2}(t,f)$, is basically just a heavily-filtered WVD with the kernel defined through a sensing matrix, $\boldsymbol{\phi}(\nu,\tau)$, thus it will suffer from the same resolution loss with which the QTFDs have to deal. This is because the ℓ_2 norm measures a signal energy, and it is not a sparsity inducing function.

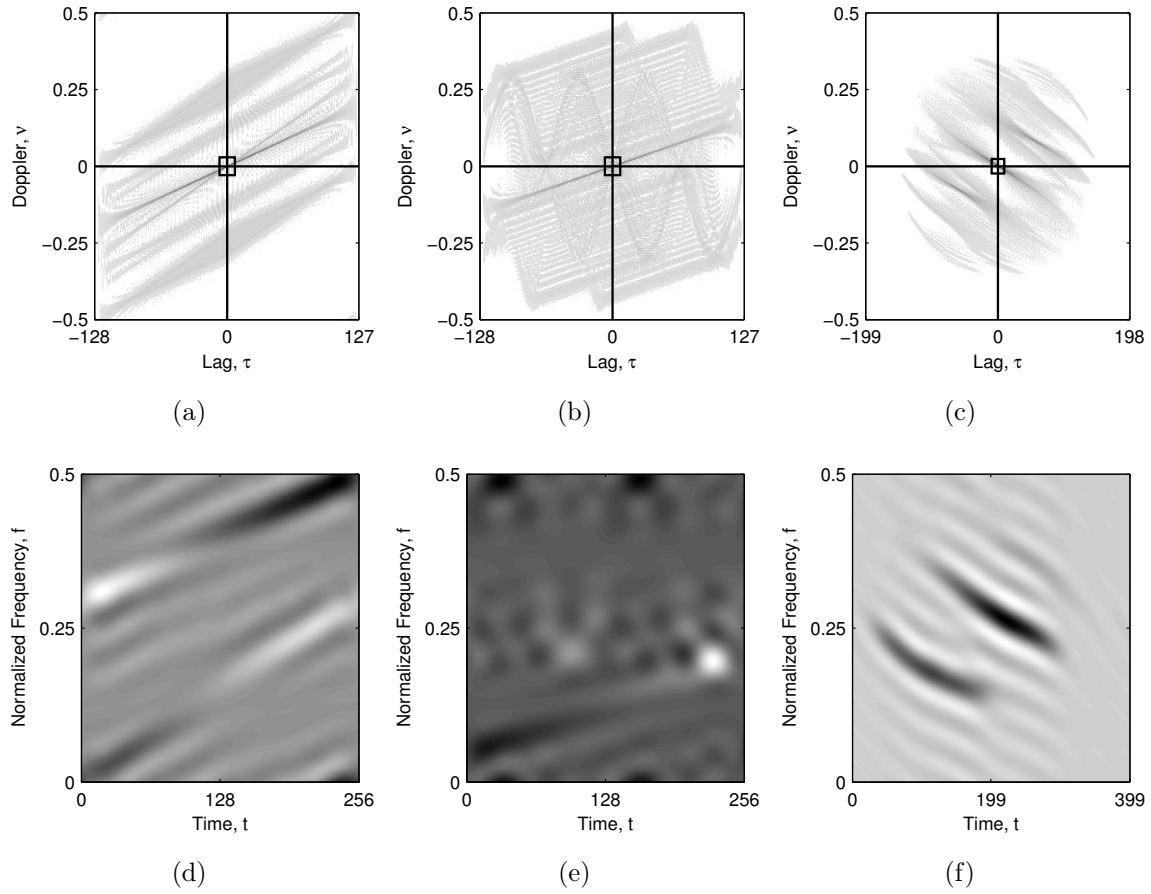


Figure 3.2: The reconstructed sparse TFDs based on the ℓ_2 norm minimization: (a) - (c) CS-AF of the signals $z_{\mathrm{LFM}}(t)$ ($N'_\tau = N'_\nu = 15$), $z_{\mathrm{nonLFM}}(t)$ ($N'_\tau = N'_\nu = 15$), and $z_{\mathrm{bat}}(t)$ ($N'_\tau = N'_\nu = 19$), (d) - (f) the reconstructed sparse TFDs of the signals $z_{\mathrm{LFM}}(t)$ ($R_z^{\alpha_{\mathrm{R}}}\big|_{\alpha_{\mathrm{R}}=3} = 8.4958$, and $M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 11.1723$), $z_{\mathrm{nonLFM}}(t)$ ($R_z^{\alpha_{\mathrm{R}}}\big|_{\alpha_{\mathrm{R}}=3} = 8.4856$, and $M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 10.6073$), and $z_{\mathrm{bat}}(t)$ ($R_z^{\alpha_{\mathrm{R}}}\big|_{\alpha_{\mathrm{R}}=3} = 13.1724$, and $M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 10.6056$).

3.3.3 ℓ_0 Norm Based TFD Reconstruction

Unlike the regularization function based on the ℓ_2 norm, described in Section 3.3.2, the ℓ_0 norm based regularization function induces sparsity by counting a number of nonzero elements of the $\boldsymbol{\vartheta}_z(t,f)$, hence can be written as:

$$c^{\ell_0}\left(\boldsymbol{\vartheta}_z(t,f)\right) = ||\boldsymbol{\vartheta}_z(t,f)||_0 = \sum^{N_t}\sum^{N_f} |\boldsymbol{\vartheta}_z(t,f)|^0 = \sum_{\boldsymbol{\vartheta}_z(t,f)\neq 0}^{N_t}\sum^{N_f} 1, \tag{3.15}$$

thus the optimization problem (3.11) becomes:

$$\begin{aligned} \boldsymbol{\vartheta}_z^{\ell_0}(t,f) &= \arg\min_{\boldsymbol{\vartheta}_z(t,f)} ||\boldsymbol{\vartheta}_z(t,f)||_0\,, \\ &\text{subject to: } \left|\left|\boldsymbol{\vartheta}_z(t,f) - \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau)\right|\right|_2^2 \leq \epsilon. \end{aligned} \tag{3.16}$$

Solution of the minimization problem (3.16) has a unique and a well known closed form:

$$\boldsymbol{\vartheta}_z^{\ell_0}(t,f) = \text{hard}_{\sqrt{2\lambda}}\left\{\boldsymbol{\vartheta}_z(t,f)\right\}, \tag{3.17}$$

where the operator $\text{hard}_{\sqrt{2\lambda}}\left\{\boldsymbol{\vartheta}_z(t,f)\right\}$ is a hard-thresholding function, defined as:

$$\text{hard}_{\sqrt{2\lambda}}\left\{\boldsymbol{\vartheta}_z(t,f)\right\} = \begin{cases} \boldsymbol{\vartheta}_z(t,f), & |\boldsymbol{\vartheta}_z(t,f)| > \sqrt{2\lambda}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.18}$$

It is well known that the problem of ℓ_0 norm minimization is computationally NP hard [3, 61, 59, 13]; however, recently a several iterative algorithms have been developed addressing this problem. Sparse signal reconstruction algorithms based on the ℓ_0 norm minimization used in this thesis are shortly discussed in the sequel of this section.

- Iterative hard thresholding (IHT)[1] algorithm iterations are defined as [14]:

$$\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n+1]} = H_K\left\{\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]} + \mu_{\text{IHT}}\boldsymbol{\psi}^H\left(\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]}\right)\right\}, \tag{3.19}$$

  where the parameter μ_{IHT} defines the algorithm step size, $[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)]^{[n]}$ denotes the n-th algorithm iteration, the starting solution is initialized as $[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)]^{[0]} = \boldsymbol{\psi}^H A'_z(\nu,\tau)$, and $H_K\{\cdot\}$ denotes a non-linear operator which keeps the largest K samples, by setting all of the remaining samples to zero. Note that the operator H_K is basically a hard-thresholding function (3.18) defined in a different way. The IHT algorithm step size, μ_{IHT}, can be adaptively calculated, leading to the normalized IHT (NIHT) algorithm [13]:

$$\mu_{\text{NIHT}} = \frac{\left|\left|\boldsymbol{\psi}^H_{\Gamma^{[n]}}\left(\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]}\right)\right|\right|_2^2}{\left|\left|\boldsymbol{\psi}_{\Gamma^{[n]}}\left(\boldsymbol{\psi}^H\left(\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]}\right)\right)\right|\right|_2^2}, \tag{3.20}$$

  where $\Gamma^{[n]}$ are indexes of the non-zero elements of $\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]}$, that is $\Gamma^{[n]} = \text{supp}\left(\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]}\right)$, while $\boldsymbol{\psi}_{\Gamma^{[n]}}$ is a matrix with elements outside the set $\Gamma^{[n]}$ removed. Furthermore, the IHT convergence rate can be accelerated, leading to the accelerated IHT (AIHT) algorithm [13]. This is achieved by calculating $\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]}$ with (3.19); however, before proceeding to the next iteration of the IHT algorithm, the AIHT algorithm tries to find an estimate $\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n+1]}$ which

[1] Available at: `http://www.personal.soton.ac.uk/tb1m08/sparsify/sparsify_0_5.zip`

satisfies the following two conditions:

1. $\left|\left|\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n+1]}\right|\right|_0 = K,$ (3.21a)
2. $\left|\left|\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n+1]}\right|\right|_2^2 \leq \left|\left|\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]}\right|\right|_2^2.$ (3.21b)

- Expectation-Conditional Maximisation Either (ECME)[2] algorithm replaces the IHT algorithm step size μ_{IHT} with a pseudo-inverse of $\boldsymbol{\psi}$, that is $\mu_{\text{ECME}} = \left(\boldsymbol{\psi}\boldsymbol{\psi}^H\right)^{-1}$, leading to the [58]:

$$\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n+1]} = H_K\left\{\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]} + \boldsymbol{\psi}^H(\boldsymbol{\psi}\boldsymbol{\psi}^H)^{-1}\left(\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]}\right)\right\}. \quad (3.22)$$

  Calculation of μ_{ECME} in this way, removes the problem of step size selection, which can cause instability of the IHT algorithm. Note, that the ECME algorithm reduces to the IHT algorithm with $\mu_{\text{IHT}} = 1$ when $(\boldsymbol{\psi}\boldsymbol{\psi}^H)^{-1} = 1$.

- Double overrelaxation thresholding scheme (DORE)[2] algorithm calculates $\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]}$ with (3.19), just like the IHT algorithm; however, it is followed by the two over-relaxation steps [59]:

$$\left[\widetilde{\boldsymbol{\vartheta}}_{z1}^{\ell_0}(t,f)\right]^{[n+1]} = \left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]} + a_{1_{\text{DORE}}}\left(\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]} - \left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n]}\right), \quad (3.23a)$$

$$\left[\widetilde{\boldsymbol{\vartheta}}_{z2}^{\ell_0}(t,f)\right]^{[n+1]} = H_K\left\{\left[\widetilde{\boldsymbol{\vartheta}}_{z1}^{\ell_0}(t,f)\right]^{[n+1]} + a_{2_{\text{DORE}}}\left(\left[\widetilde{\boldsymbol{\vartheta}}_{z1}^{\ell_0}(t,f)\right]^{[n+1]} - \left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n-1]}\right)\right\}, \quad (3.23b)$$

  where $a_{1_{\text{DORE}}}$ and $a_{2_{\text{DORE}}}$ are the line search parameters, and a solution of the $[n+1]$-th algorithm iteration, $\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n+1]}$, is selected as $\left[\widetilde{\boldsymbol{\vartheta}}_{z2}^{\ell_0}(t,f)\right]^{[n+1]}$ if the following condition is fulfilled:

$$\left|\left|\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\widetilde{\boldsymbol{\vartheta}}_{z2}^{\ell_0}(t,f)\right]^{[n+1]}\right|\right|_2^2 \leq \left|\left|\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]}\right|\right|_2^2, \quad (3.24)$$

  and $\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]}$ otherwise. If the signal sparsity level K is unknown, the automatic DORE (ADORE) algorithm can be used which utilizes the unconstrained sparsity selection (USS) criterion [59].

- Hard thresholding pursuit (HTP)[3] algorithm uses the IHT algorithm iteration (3.19) to find a preliminary solution, which is then refined by solving the following opti-

[2] ECME and DORE algorithms are available at: `http://home.eng.iastate.edu/~ald/ECME.tar.gz`
[3] Available at: `https://github.com/foucart/HTP`

mization problem [32]:

$$\left[\boldsymbol{\vartheta}_z^{\ell_0}(t,f)\right]^{[n+1]} = \underset{\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]}}{\arg\min} \left|\left|\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}_{\Gamma^{[n+1]}}\left[\widetilde{\boldsymbol{\vartheta}}_z^{\ell_0}(t,f)\right]^{[n+1]}\right|\right|_2^2, \tag{3.25}$$

with the conjugate gradient algorithm. In [32] the normalized (NHTP) version and the fast (FHTP) version of the HTP algorithm have been derived.

Example 3.2. In this example, the reconstructed sparse TFDs based on the ℓ_0 norm minimization have been calculated for the signals $z_{\mathrm{LFM}}(t)$, $z_{\mathrm{nonLFM}}(t)$, and $z_{\mathrm{bat}}(t)$, with the following algorithms: IHT_λ, IHT_K, NIHT_λ, NIHT_K, AIHT_K, ECME_K, DORE_K, HTP_K, and FHTP_K. Note that the subscript λ in the algorithm names denotes that the hard-thresholding function is implemented with the hard operator, while the subscript K denotes that the hard-thresholding function is implemented with the H_K operator. The expected sparsity level has been calculated as $K = 5N_cN_t$ (N_c is used as the prior information about the signal), the regularization parameter has been selected as $\lambda = 10$, while the CS-AF area has been selected as in Example 3.1 and is shown in Figs. 3.2(a) - 3.2(c). The resulting reconstructed sparse TFDs are shown in Figs. 3.3 - 3.5 with the results summarized in Table 3.1 for all the tested signals and algorithms. Note that the IHT_λ algorithm and the NIHT_λ algorithm for the signal $z_{\mathrm{bat}}(t)$ did not converge, producing NaN solution, and this is why the respective sparse reconstruction results have been omitted from Fig. 3.5 and Table 3.1.

The algorithm execution times are obtained on the PC with the Intel Core i7-4770 @ 3.40 Ghz processor and 16 GB of RAM, and should be taken only in comparison among themselves. By visual inspection of the presented results, it can be seen that

Table 3.1: Rényi entropies, $R_z^{\alpha_\mathrm{R}}$, concentration measures, M_z^S, and the algorithm execution times of the reconstruction algorithms based on the ℓ_0 norm minimization. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	$R_z^{\alpha_\mathrm{R}}$ $\alpha_\mathrm{R}=3$	M_z^S $p_\mathrm{S}=2$	t $[ms]$	$R_z^{\alpha_\mathrm{R}}$ $\alpha_\mathrm{R}=3$	M_z^S $p_\mathrm{S}=2$	t $[ms]$	$R_z^{\alpha_\mathrm{R}}$ $\alpha_\mathrm{R}=3$	M_z^S $p_\mathrm{S}=2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IHT_λ	13.0632	0.1534	38.8363	12.0882	0.0757	70.8456	–	–	–
IHT_K	11.8760	0.0584	141.514	11.2558	0.0388	272.996	12.7482	0.0489	215.227
NIHT_λ	12.7974	0.1296	35.5623	12.0462	0.0722	21.2055	–	–	–
NIHT_K	11.5585	0.0569	174.825	11.1379	0.0383	88.4535	**12.6429**	0.0479	285.033
AIHT_K	**10.1025**	0.0493	3317.7	10.0377	0.0328	2696.0	12.7620	0.0489	39.0654
ECME_K	10.4123	**0.0491**	911.618	10.0114	0.0334	916.207	12.7571	0.0489	64.5025
DORE_K	10.1620	0.0487	2009.8	**9.5909**	**0.0316**	2150.3	12.6548	**0.0475**	96.1028
HTP_K	11.8785	0.0584	70.6926	11.0575	0.0382	29.1090	12.7416	0.0488	75.9146
FHTP_K	14.2817	0.7253	**14.5512**	14.1350	0.7353	**12.5697**	14.5039	0.5680	**32.2825**

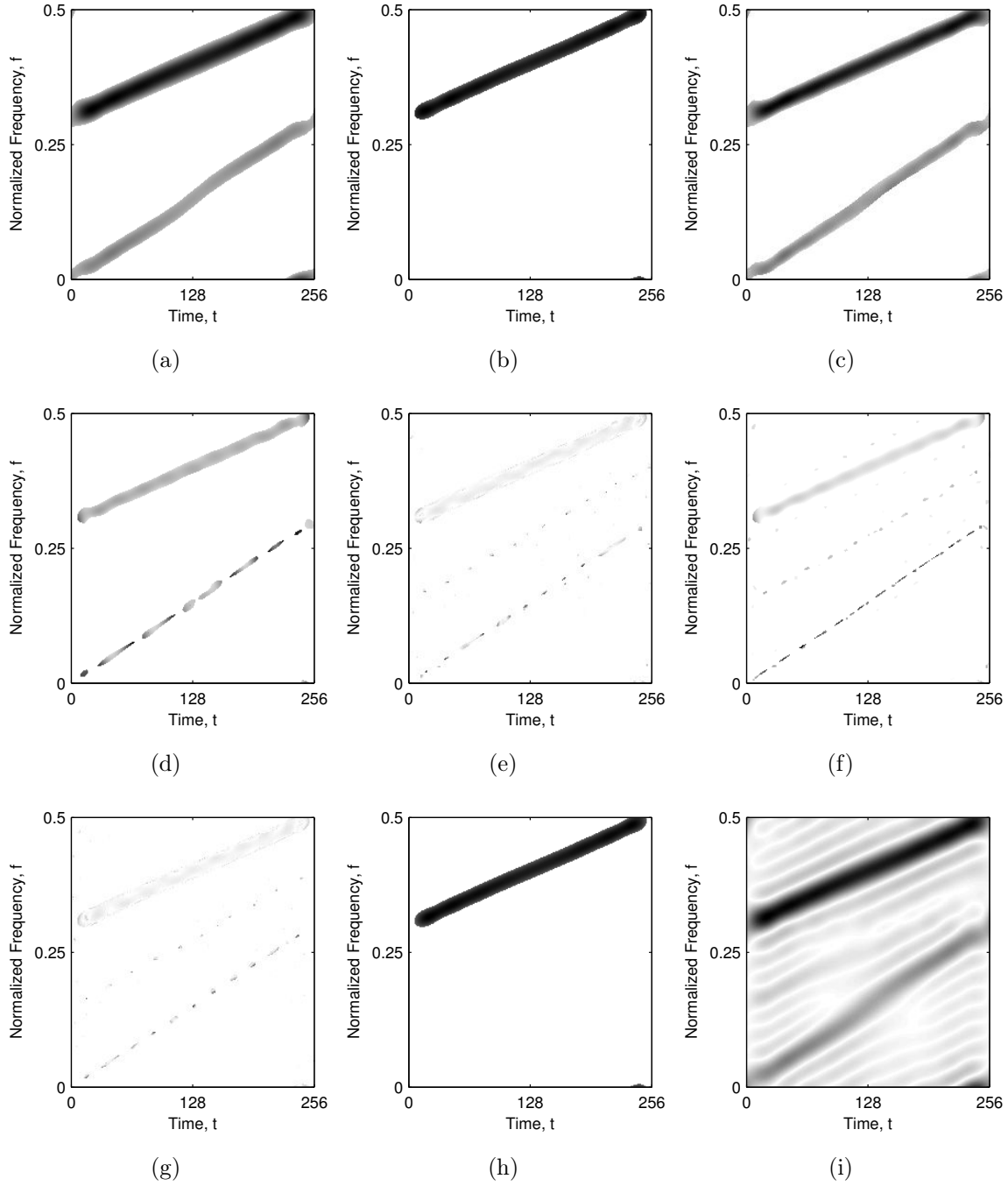


Figure 3.3: The reconstructed sparse TFDs based on the ℓ_0 norm minimization of the three LFM component signal $z_{\mathrm{LFM}}(t)$ with: (a) IHT_{λ}, (b) IHT_{K}, (c) NIHT_{λ}, (d) NIHT_{K}, (e) AIHT_{K}, (f) ECME_{K}, (g) DORE_{K}, (h) HTP_{K}, (i) FHTP_{K} algorithm.

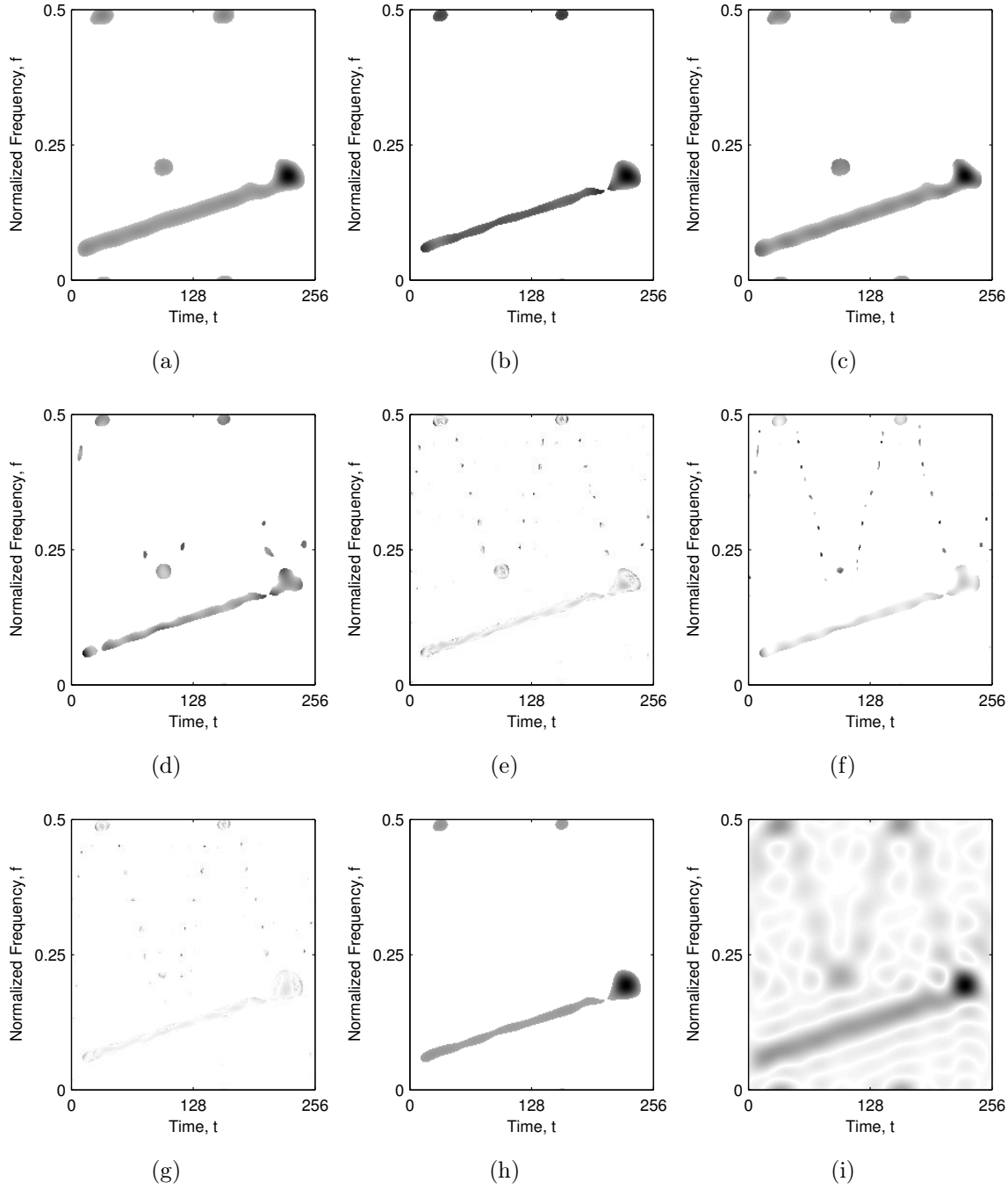


Figure 3.4: The reconstructed sparse TFDs based on the ℓ_0 norm minimization of the two component signal $z_{\mathrm{nonLFM}}(t)$ with: (a) IHT_{λ}, (b) IHT_{K}, (c) NIHT_{λ}, (d) NIHT_{K}, (e) AIHT_{K}, (f) ECME_{K}, (g) DORE_{K}, (h) HTP_{K}, (i) FHTP_{K} algorithm.

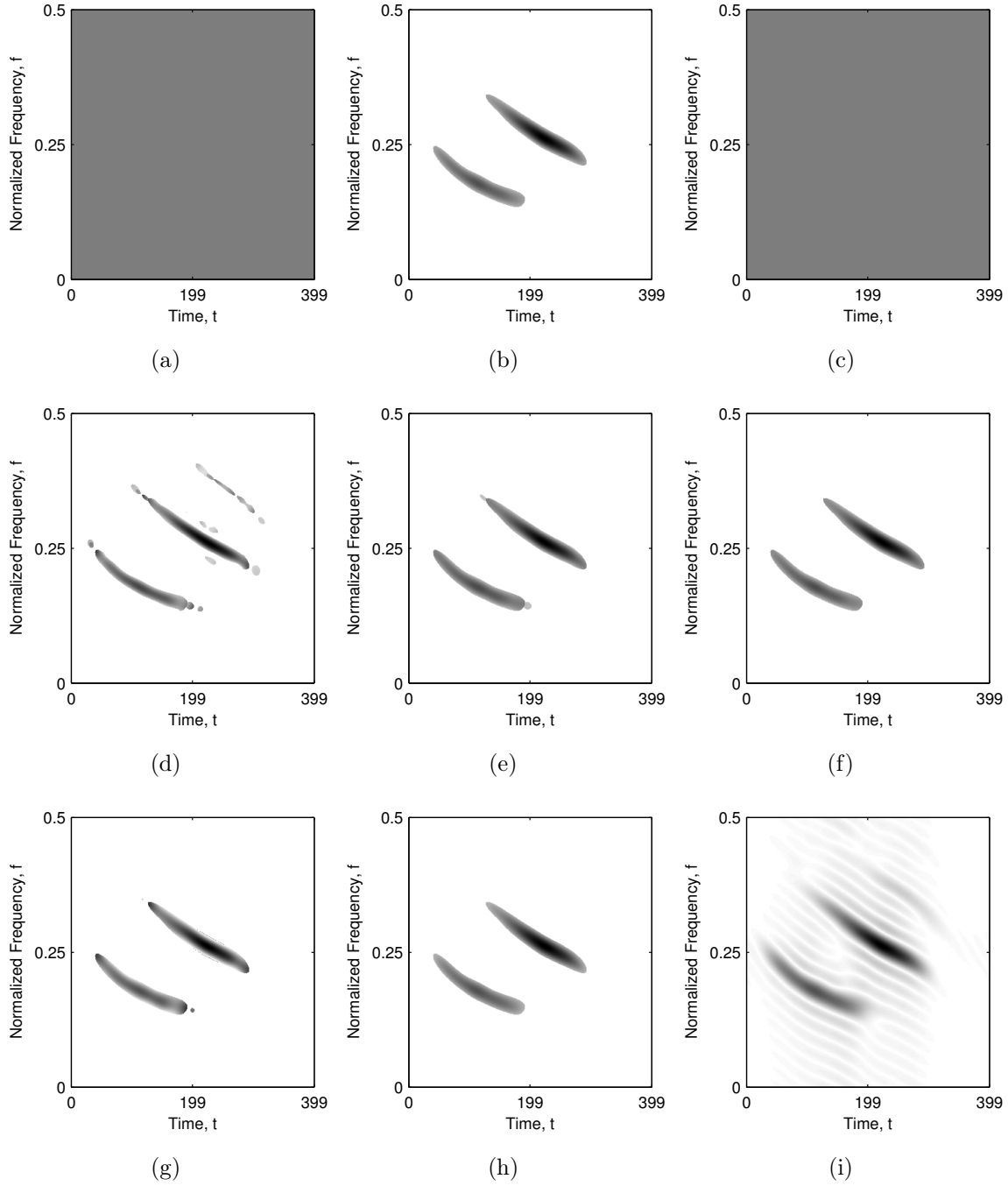


Figure 3.5: The reconstructed sparse TFDs based on the ℓ_0 norm minimization of the bat echolocation signal $z_{\text{bat}}(t)$ with: (a) IHT_λ, (b) IHT_K, (c) NIHT_λ, (d) NIHT_K, (e) AIHT_K, (f) ECME_K, (g) DORE_K, (h) HTP_K, (i) FHTP_K algorithm.

most of the ℓ_0 norm based sparse TFDs are oversparse, as the auto-terms do not form a single continuous line. Furthermore, most of the tested algorithms have filtered out the weakest component, leaving only poorly localized stronger components. The algorithms in which the hard-thresholding function is implemented with the H_K operator perform better then the ones with the hard operator. By comparing the concentration measures, with the ones previously obtained for the classically filtered TFDs, shown in Table 2.3, it can be concluded that the obtained sparse TFDs are significantly better concentrated; however, this is not necessarily the case, since the lower concentration measures in this example have been influenced by the missing low-energy components, rather then the higher concentrated auto-terms.

The main problem of algorithms based on the ℓ_0 norm minimization is that their solution, $\boldsymbol{\vartheta}_z^{\ell_0}(t,f)$, is highly dependant on value of the input hard-thresholding parameter K (if the hard-thresholding is implemented with the H_K operator) or λ (if the hard-thresholding is implemented with the hard operator). The H_K operator requires the prior knowledge of the signal sparsity level K, which is in TFD context proportional to the number of signal components, that is $K \sim N_c N_t$. This information is often not accessible, and this is one of the reasons why algorithms based on the ℓ_0 norm minimization are in general, not used in the TFD related applications. The second reason why the ℓ_0 norm based sparse TFDs are not used, is a problem demonstrated in Example 3.2, that is, inability to resolve the low-energy components properly. Both of the discussed problems can be partially solved by estimating a TFD sparsity level by using an information about the instantaneous number of signal components calculated by the localized Rényi entropy introduced in [71, 72], as it is discussed in Section 5.3.

3.3.4 ℓ_1 Norm Based TFD Reconstruction

It has been shown that the regularization function based on the ℓ_1 norm gives the best result among all the previously discussed regularization functions, since the ℓ_1 norm provides a compromise between the sparsity inducing ℓ_0 norm and the energy measuring ℓ_2 norm [21, 3]. The regularization function based on the ℓ_1 norm sums all samples of the TFD, that is:

$$c^{\ell_1}(\boldsymbol{\vartheta}_z(t,f)) = ||\boldsymbol{\vartheta}_z(t,f)||_1 = \sum^{N_t}\sum^{N_f} |\boldsymbol{\vartheta}_z(t,f)|\,, \tag{3.26}$$

and the optimization problem (3.11) becomes:

$$\begin{aligned} \boldsymbol{\vartheta}_z^{\ell_1}(t,f) = \arg\min_{\boldsymbol{\vartheta}_z(t,f)} ||\boldsymbol{\vartheta}_z(t,f)||_1\,, \\ \text{subject to: } \left|\left|\boldsymbol{\vartheta}_z(t,f) - \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau)\right|\right|_2^2 \leq \epsilon. \end{aligned} \tag{3.27}$$

In a similar way to the ℓ_0 norm minimization solution (3.17), solution of the problem (3.27) has also a unique and a well known closed form:

$$\boldsymbol{\vartheta}_z^{\ell_1}(t,f) = \text{soft}_\lambda \left\{ \boldsymbol{\vartheta}_z(t,f) \right\}, \tag{3.28}$$

where the operator $\text{soft}_\lambda \left\{ \boldsymbol{\vartheta}_z(t,f) \right\}$ is a soft-thresholding function defined as:

$$\text{soft}_\lambda \left\{ \boldsymbol{\vartheta}_z(t,f) \right\} = \text{sgn}\left(\boldsymbol{\vartheta}_z(t,f) \right) \max\left(|\boldsymbol{\vartheta}_z(t,f)| - \lambda, 0 \right). \tag{3.29}$$

The ℓ_1 based minimization has been a popular topic of research in the last decade, resulting in an overwhelming number of methods and software packages for solving this optimization problem in various ways. The full and comprehensive list of methods is beyond the scope of this thesis; however, in continuation of this section a list of the optimization algorithms used in this thesis has been given, with a short description and links to the original MATLAB function. Note however, that some of the sparse reconstruction algorithms had to be slightly modified in order to work with the given problem formulation.

- Iterative shrinkage/thresholding (IST)[4] algorithm has the form of [25]:

$$\begin{aligned} \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n+1]} &= (1 - \beta_{\text{IST}}) \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n]} + \\ &+ \beta_{\text{IST}} \, \text{soft}_\lambda \left\{ \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n]} + \boldsymbol{\psi}^H \left(\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi} \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n]} \right) \right\}, \end{aligned} \tag{3.30}$$

  where $\beta_{\text{IST}} > 0$ is the relaxation parameter defining a trade-off between the IST algorithm accuracy and convergence rate. The original IST algorithm has the form with $\beta_{\text{IST}} = 1$; however, choosing $\beta_{\text{IST}} \neq 1$ will produce the under-relaxed IST algorithm for $\beta_{\text{IST}} < 1$, or the over-relaxed IST algorithm for $\beta_{\text{IST}} > 1$.

- Two-step IST (TwIST)[5] algorithm is very similar to the IST algorithm; however, the iterative solution is calculated from solution of the previous two iterations, rather then just one [12]:

$$\begin{aligned} \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n+1]} &= (1 - \alpha_{\text{TWIST}}) \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n-1]} + (\alpha_{\text{TWIST}} - \beta_{\text{TWIST}}) \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n]} + \\ &+ \beta_{\text{TWIST}} \, \text{soft}_\lambda \left\{ \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n]} + \boldsymbol{\psi}^H \left(\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi} \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f) \right]^{[n]} \right) \right\}, \end{aligned} \tag{3.31}$$

  where $0 < \alpha_{\text{TWIST}} \leq 1$ and $0 < \beta_{\text{TWIST}} < 2\alpha_{\text{TWIST}}$ are the relaxation parameters. When compared to the original IST algorithm, the TwIST algorithm handles ill-posed problems more efficiently, and furthermore, it has been shown that its convergence rate is the two orders of magnitude faster.

[4] Available at: `http://www.lx.it.pt/~mtf/GPSR/GPSR_6.0.zip`
[5] Available at: `http://www.lx.it.pt/~bioucas/code/TwIST_v2.zip`

- Fast IST algorithm (FISTA)[6] decreases a computational cost of the IST algorithm from $\mathcal{O}(1/n)$ to $\mathcal{O}(1/n^2)$ by applying the accelerated gradient algorithm [10]. The FISTA algorithm is defined as:

$$\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} = \text{soft}_\lambda\left\{[\mathbf{y}(t,f)]^{[n]} - \boldsymbol{\psi}^H\left(\boldsymbol{\psi}\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \mathbf{A}'_z(\nu,\tau)\right)\right\}, \tag{3.32a}$$

$$[t]^{[n+1]} = \frac{1+\sqrt{1+4\left([t]^{[n]}\right)^2}}{2}, \tag{3.32b}$$

$$[\mathbf{y}(t,f)]^{[n+1]} = \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} + \frac{[t]^{[n]}-1}{[t]^{[n+1]}}\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} - \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right). \tag{3.32c}$$

  Note that the soft-thresholding operator is applied to $[\mathbf{y}(t,f)]^{[n]}$, which is calculated as a linear combination of the previous two solutions, similarly to the TwIST algorithm.

- Sparse reconstruction by separable approximation (SpaRSA)[7] framework [82] gradually decreases a thresholding parameter across the two levels of algorithm iterations. The thresholding parameter $[\alpha_{\text{SPARSA}}]^{[t+1]}$, in the outer algorithm iterations is updated according to the Barzilai-Borwein (BB) spectral approach:

$$[\alpha_{\text{SPARSA}}]^{[t+1]} = \frac{\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n-1]}\right)^T\left(\nabla f\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right) - \nabla f\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n-1]}\right)\right)}{\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n-1]}\right)^T\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n-1]}\right)}, \tag{3.33}$$

  where $\nabla f\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n-1]}\right)$ is the gradient of the error estimating function, defined in (3.11). On the other hand, the inner algorithm iterations calculate the solution, $\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]}$, as:

$$\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} = \text{soft}_{\frac{\lambda}{[\alpha_{\text{SPARSA}}]^{[t+1]}}}\left\{\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \frac{1}{[\alpha_{\text{SPARSA}}]^{[t+1]}}\nabla f\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right)\right\}, \tag{3.34}$$

  where $[\alpha_{\text{SPARSA}}]^{[t+1]}$ in the each inner algorithm iteration is decreased with the user defined parameter $\eta_{\text{SPARSA}} < 1$ as: $[\alpha_{\text{SPARSA}}]^{[t+1]} \leftarrow \eta_{\text{SPARSA}}[\alpha_{\text{SPARSA}}]^{[t+1]}$, until the acceptance criterium is satisfied. The described algorithm structure is often called the continuation scheme, which can be implemented in different ways. The general characteristics of all algorithms which utilize the continuation scheme is their faster convergence rate, since the thresholding parameter, $\lambda/[\alpha_{\text{SPARSA}}]^{[t+1]}$, is continuously updated.

- Fixed point continuation (FPC)[8] algorithm is based on the forward-backward splitting method combined with the continuation scheme [36]. Through a forward-

[6] Available at: `http://web.iem.technion.ac.il/images/user-files/becka/papers/wavelet_FISTA.zip`
[7] Available at: `http://www.lx.it.pt/~mtf/SpaRSA/SpaRSA_2.0.zip`
[8] Available at: `http://www.caam.rice.edu/~optimization/L1/fpc/fpc.zip`

backward splitting algorithm, the FPC algorithm is derived as:

$$\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} = \text{soft}_{\frac{\lambda}{[\mu_{\text{FPC}}]^{[n+1]}}}\left\{\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \tau_{\text{FPC}}\nabla f\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right)\right\}, \quad (3.35)$$

which is equivalent to the SpaRSA algorithm. The difference between the algorithms lies in the way in which the continuation scheme is implemented. The parameter τ_{FPC} is calculated as $\tau_{\text{FPC}} = \min\left(1 + 1.665\left(1 - \frac{N'_\nu N'_\tau}{N_t N_f}\right), 1.9999\right)$, from which μ_{FPC} is initialized as $[\mu_{\text{FPC}}]^{[0]} = \frac{\tau_{\text{FPC}}}{\gamma_{\text{FPC}}}\left|\left|\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[0]}\right|\right|_\infty^{-1}$, where $0 < \gamma_{\text{FPC}} < 1$ is the user defined parameter. The continuation scheme is implemented in the outer algorithm iterations by increasing value of the parameter $[\mu_{\text{FPC}}]^{[n+1]} = \beta_{\text{FPC}}[\mu_{\text{FPC}}]^{[n]}$, where $\beta_{\text{FPC}} > 1$ is the user defined parameter.

- Gradient projection for sparse reconstruction (GPSR)[9] algorithm [29] transforms the optimization problem (3.11) to a quadratic optimization problem, by splitting the solution into two parts: $\mathbf{u}(t,f)$ and $\mathbf{v}(t,f)$, that is: $\boldsymbol{\vartheta}_z^{\ell_1}(t,f) = \mathbf{u}(t,f) - \mathbf{v}(t,f)$, where $\mathbf{u}(t,f) = \max\left(0, \boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right)$ and $\mathbf{v}(t,f) = \max\left(0, -\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right)$, resulting in the optimization problem which is easier to solve. This equivalent optimization problem is then solved by the projected gradient steps combined with the BB method in order to accelerate the algorithm convergence rate. With the notation $\mathbf{z}(t,f) = \left[\mathbf{u}^T(t,f) \;\; \mathbf{v}^T(t,f)\right]^T$, the GPSR algorithm iterations are defined as:

$$[\boldsymbol{\delta}(t,f)]^{[n+1]} = \max\left(0, [\mathbf{z}(t,f)]^{[n]} - [\alpha]^{[n]}\nabla f\left([\mathbf{z}(t,f)]^{[n]}\right)\right) - [\mathbf{z}(t,f)]^{[n]}, \quad (3.36a)$$

$$[\lambda]^{[n+1]} = \min\left(1, \max\left(0, \frac{\left([\boldsymbol{\delta}(t,f)]^{[n+1]}\right)^T \nabla f\left([\mathbf{z}(t,f)]^{[n]}\right)}{\left([\boldsymbol{\delta}(t,f)]^{[n+1]}\right)^T [\boldsymbol{\delta}(t,f)]^{[n+1]}}\right)\right), \quad (3.36b)$$

$$[\mathbf{z}(t,f)]^{[n+1]} = [\mathbf{z}(t,f)]^{[n]} + [\lambda]^{[n+1]}[\boldsymbol{\delta}(t,f)]^{[n+1]}, \quad (3.36c)$$

$$[\gamma]^{[n+1]} = \left([\boldsymbol{\delta}(t,f)]^{[n+1]}\right)^T [\boldsymbol{\delta}(t,f)]^{[n+1]}, \quad (3.36d)$$

$$[\alpha]^{[n+1]} = \min\left(\alpha_{\max}, \max\left(\alpha_{\min}, \frac{\left|\left|[\boldsymbol{\delta}(t,f)]^{[n+1]}\right|\right|_2^2}{[\gamma]^{[n+1]}}\right)\right). \quad (3.36e)$$

The main problem of all methods based on the gradient projection is their performance deterioration when the regularization function is undervalued (e.g. value of the parameter λ is very low); however, the GPSR algorithm employs a continuation scheme in order to bypass this problem.

- Nestrov algorithm (NESTA)[10] is based on the smoothing algorithm developed by

[9] Available at: `http://www.lx.it.pt/~mtf/GPSR/GPSR_6.0.zip`
[10] Available at: `http://statweb.stanford.edu/~candes/nesta/NESTA_v1.1.zip`

Nesterov [11], and it is defined as:

$$\begin{aligned}[\mathbf{y}(t,f)]^{[n+1]} &= \left(\mathbf{I} - \frac{\lambda_\epsilon}{\lambda_\epsilon + \lambda^{-1}}\right) \\ &\quad \left(\lambda_\epsilon \lambda \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau) + \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \lambda \nabla f_\lambda \left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right)\right), \end{aligned} \tag{3.37a}$$

$$\begin{aligned}[\mathbf{z}(t,f)]^{[n+1]} &= \left(\mathbf{I} - \frac{\lambda_\epsilon}{\lambda_\epsilon + \lambda^{-1}}\right) \\ &\quad \left(\lambda_\epsilon \lambda \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau) + \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[0]} - \lambda \sum_{i \le n} [\alpha_{\text{NESTA}}]^{[i]} \nabla f_\lambda \left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[i]}\right)\right), \end{aligned} \tag{3.37b}$$

$$\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} = [\tau_{\text{NESTA}}]^{[n+1]} [\mathbf{z}(t,f)]^{[n+1]} + (1 - [\tau_{\text{NESTA}}]^{[n+1]}) [\mathbf{y}(t,f)]^{[n+1]}, \tag{3.37c}$$

where the parameters $[\alpha_{\text{NESTA}}]^{[n]}$ and $[\tau_{\text{NESTA}}]^{[n]}$ are defined as: $[\alpha_{\text{NESTA}}]^{[n]} = 1/2(n+1)$ and $[\tau_{\text{NESTA}}]^{[n]} = 2/(n+3)$, respectively. Furthermore, the optimal Lagrange multiplier, λ_ϵ, can be explicitly calculated in the closed form as:

$$\lambda_\epsilon = \max\left(0, \frac{\left\| \mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} - \lambda \nabla f_\lambda \left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]}\right)\right)\right\|_2}{\epsilon} - \frac{1}{\lambda}\right), \tag{3.38}$$

where $\nabla f_\lambda \left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right)$, gradient of the Huber function, is also given in its closed form as:

$$\nabla f_\lambda \left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right) = \begin{cases} \dfrac{1}{\lambda}\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}, & \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} < \lambda, \\ \operatorname{sgn}\left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right), & \text{otherwise.} \end{cases} \tag{3.39}$$

In a similar way to the FISTA algorithm, the key idea of the NESTA algorithm is to use the solutions from the previous algorithm iterations in the soft-thresholding operator. In the NESTA algorithm this is implemented in (3.37b), where $[\mathbf{z}(t,f)]^{[n+1]}$ is calculated by averaging a weighted sum of gradients from the previous algorithm iterations, which has been shown to increase the convergence rate of the NESTA algorithm, making it appealing for the large-scale problems.

- Split augmented Lagrangian shrinkage algorithm (SALSA)[11] transforms the unconstrained optimization problem (3.27) into an equivalent constrained optimization problem by the means of variable splitting [2]: $\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} = [\mathbf{v}(t,f)]^{[n]} + [\mathbf{d}(t,f)]^{[n]}$. The resulting constrained problem is then solved by the alternating direction method of multipliers (ADMM), a variation of the augmented Lagrangian (AL) method. The

[11] Available at: `http://cascais.lx.it.pt/~mafonso/SALSA_v2.0.zip`

SALSA algorithm is defined as:

$$\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} = \boldsymbol{\psi}^H\left(\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\left([v(t,f)]^{[n]} + [\mathbf{d}(t,f)]^{[n]}\right)\right), \tag{3.40a}$$

$$[\mathbf{v}(t,f)]^{[n+1]} = \text{soft}_\lambda\left\{\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} - [\mathbf{d}(t,f)]^{[n]}\right\}, \tag{3.40b}$$

$$[\mathbf{d}(t,f)]^{[n+1]} = [\mathbf{d}(t,f)]^{[n]} - \left(\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} - [\mathbf{v}(t,f)]^{[n+1]}\right). \tag{3.40c}$$

The SALSA algorithm has been shown to have a fast convergence rate in practice, which is combined with a flexibility for solving the various optimization formulations and effectiveness in the large-scale non-smooth optimization problems, the characteristics which all of the ADMM based methods exhibit.

- Your augmented Lagrangian algorithm for ℓ_1 (YALL1)[12], just like SALSA algorithm is derived from the ADMM application to the AL function [85, 83]:

$$[\mathbf{y}(\nu,\tau)]^{[n+1]} = \alpha_{\text{YALL}}\boldsymbol{\psi}\,[\mathbf{z}(t,f)]^{[n]} - \beta_{\text{YALL}}\left(\boldsymbol{\psi}\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} - \mathbf{A}'_z(\nu,\tau)\right), \tag{3.41a}$$

$$[\mathbf{z}(t,f)]^{[n+1]} = \text{soft}_{\frac{\lambda}{\mu_{\text{YALL}}}}\left\{\boldsymbol{\psi}^H\,[\mathbf{y}(\nu,\tau)]^{[n+1]} + \frac{1}{\mu_{\text{YALL}}}\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right\}, \tag{3.41b}$$

$$\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} = \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} + \gamma_{\text{YALL}}\mu_{\text{YALL}}\left(\boldsymbol{\psi}^H\,[\mathbf{y}(\nu,\tau)]^{[n+1]} - [\mathbf{z}(t,f)]^{[n+1]}\right), \tag{3.41c}$$

where $\mu_{\text{YALL}} > 0$ is the penalty parameter, and the parameters α_{YALL}, β_{YALL}, and γ_{YALL} are calculated as: $\gamma_{\text{YALL}} \in (0, (1+\sqrt{5})/2)$, $\alpha_{\text{YALL}} = \mu_{\text{YALL}}/(\mu_{\text{YALL}} + \rho_{\text{YALL}})$, and $\beta_{\text{YALL}} = 1/(\mu_{\text{YALL}} + \rho_{\text{YALL}})$, respectively, with $\rho_{\text{YALL}} = 1/(2\lambda)$. The main advantages of YALL1 algorithm is its wide application and effectiveness in the various optimization problems. In terms of its performance, the main advantages of the YALL1 algorithm is its fast convergence rate, which is achieved without a continuation scheme, or a line search method (which, if used, would increase the algorithm execution time), and its robustness towards the changes of the input parameters. However, these advantages are negated when the observed signal is corrupted with the non-trivial level of noise.

- ℓ_1-ls[13] algorithm [44] is an interior point method which appends the logarithmic barrier with the bound constraint $-\boldsymbol{\varsigma}_z(t,f) \leq \boldsymbol{\vartheta}_z(t,f) \leq \boldsymbol{\varsigma}_z(t,f)$ to the objective function of the optimization problem (3.11), that is:

$$\begin{aligned}\phi_t\left(\boldsymbol{\varsigma}_z(t,f), \boldsymbol{\vartheta}_z(t,f)\right) = \frac{t_{\text{LS}}}{2}\left|\left|\boldsymbol{\vartheta}_z(t,f) - \boldsymbol{\psi}^H\mathbf{A}'_z(\nu,\tau)\right|\right|_2^2 + t_{\text{LS}}\lambda\left|\left|\boldsymbol{\varsigma}_z(t,f)\right|\right|_1 - \\ - \sum^{N_t}\sum^{N_f}\log\left(\boldsymbol{\varsigma}_z(t,f)^2 - \boldsymbol{\vartheta}_z(t,f)^2\right).\end{aligned} \tag{3.42}$$

[12] Available at:
`http://www.caam.rice.edu/~optimization/L1/YALL1/.v.beta/YALL1_Group_20120223.zip`
[13] Available at: `https://stanford.edu/~boyd/l1_ls/l1_ls_matlab.zip`

This function has a unique minimizer, producing a curve, called the central path, defined with the parameter t_{LS}. The primal interior point method minimizes (3.42) along the central path by increasing the parameter $[t_{\text{LS}}]^{[n+1]} = \mu_{\text{LS}}[t_{\text{LS}}]^{[n]}$, where the parameter μ_{LS} is usually $2 \leq \mu_{\text{LS}} \leq 50$. The minimization is performed by solving a Newton system with the precondition conjugate gradient (PCG) method:

$$\nabla^2\phi_t\left(\left[\varsigma_z^{\ell_1}(t,f)\right]^{[n]},\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right)\begin{bmatrix}\left[\Delta\varsigma_z^{\ell_1}(t,f)\right]^{[n+1]}\\ \left[\Delta\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]}\end{bmatrix} = -\nabla\phi_t\left(\left[\varsigma_z^{\ell_1}(t,f)\right]^{[n]},\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right), \tag{3.43}$$

and the solution is updated as:

$$\left[\varsigma_z^{\ell_1}(t,f)\right]^{[n+1]} = \left[\varsigma_z^{\ell_1}(t,f)\right]^{[n]} + \beta_{\text{LS}}^{k_{\text{LS}}}\left[\Delta\varsigma_z^{\ell_1}(t,f)\right]^{[n+1]}, \tag{3.44a}$$

$$\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]} = \left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} + \beta_{\text{LS}}^{k_{\text{LS}}}\left[\Delta\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]}, \tag{3.44b}$$

where the parameter $k_{\text{LS}} \geq 0$ is an integer, calculated by the backtracking line search, i.e. as the smallest integer satisfying the following inequality:

$$\begin{aligned}&\phi_t\left(\left[\varsigma_z^{\ell_1}(t,f)\right]^{[n]},\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right) + \alpha\beta_{\text{LS}}^{k_{\text{LS}}}\nabla\phi_t\left(\left[\varsigma_z^{\ell_1}(t,f)\right]^{[n]},\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]}\right)^T\begin{bmatrix}\left[\Delta\varsigma_z^{\ell_1}(t,f)\right]^{[n+1]}\\ \left[\Delta\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]}\end{bmatrix}\\ &\quad\geq \phi_t\left(\left[\varsigma_z^{\ell_1}(t,f)\right]^{[n]} + \beta_{\text{LS}}^{k_{\text{LS}}}\left[\Delta\varsigma_z^{\ell_1}(t,f)\right]^{[n+1]},\left[\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n]} + \beta_{\text{LS}}^{k_{\text{LS}}}\left[\Delta\boldsymbol{\vartheta}_z^{\ell_1}(t,f)\right]^{[n+1]}\right),\end{aligned} \tag{3.45}$$

where α_{LS} and β_{LS} are the line search parameters.

Example 3.3. In this example, the reconstructed sparse TFDs based on the ℓ_1 norm minimization have been calculated for the signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$, with the following algorithms: IST, TwIST, FISTA, SpaRSA, FPC, GPSR, NESTA, SALSA, YALL1, and ℓ_1-ls. The regularization parameter has been set as: $\lambda = 1$, while the CS-AF area has been selected as in Example 3.1 and is shown in Figs. 3.2(a) - 3.2(c). The resulting ℓ_1 norm based reconstructed sparse TFDs are shown in Figs. 3.6 - 3.8, with the results summarized in Table 3.2 for all the tested signals and algorithms. Note that the FPC algorithm for the signal $z_{\text{bat}}(t)$ did not converge, producing NaN solution, and this is why the respective sparse reconstruction result has been omitted from Table 3.2 and Fig. 3.2(c).

The algorithm execution times are obtained on the PC with the Intel Core i7-4770 @ 3.40 Ghz processor and 16 GB of RAM, and should be taken only in comparison among themselves. Note however, that the obtained results should not be taken to rigorously, since the comprehensive comparison of the ℓ_1 based sparse reconstruction algorithms is not a topic of this thesis and it would require an extensive modifications to the original algorithms, and even then it would not guaranty the objectivity of the comparison, due to the subtle differences in the algorithm implementations, differences in the termination criteria, and large number of the algorithm specific input parameters which can not be

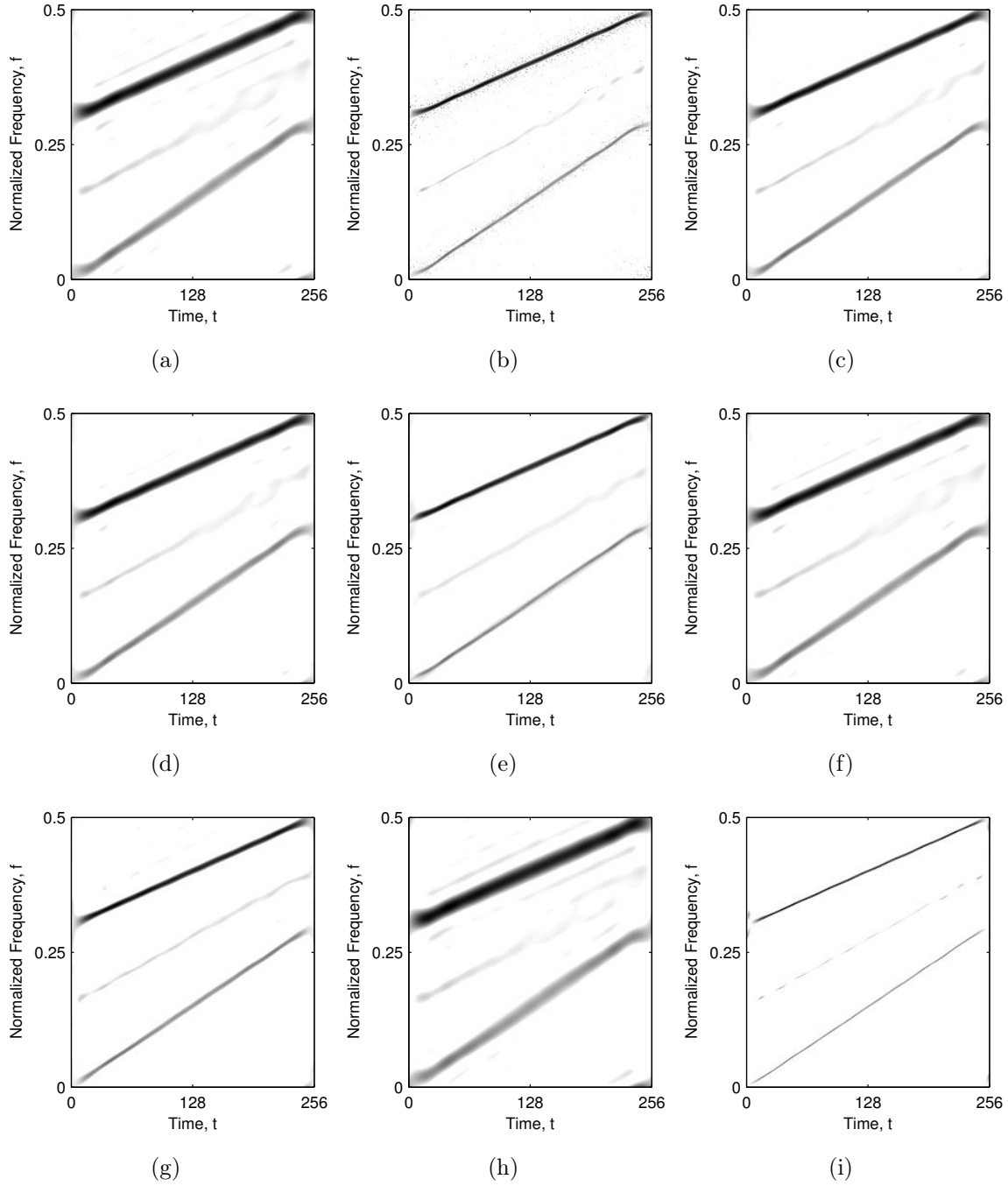


Figure 3.6: The reconstructed sparse TFDs based on the ℓ_1 norm minimization of the three LFM component signal $z_{\mathrm{LFM}}(t)$ with: (a) IST, (b) TwIST, (c) FISTA, (d) SpaRSA, (e) FPC, (f) GPSR, (g) NESTA, (h) SALSA, (i) YALL1 algorithm.

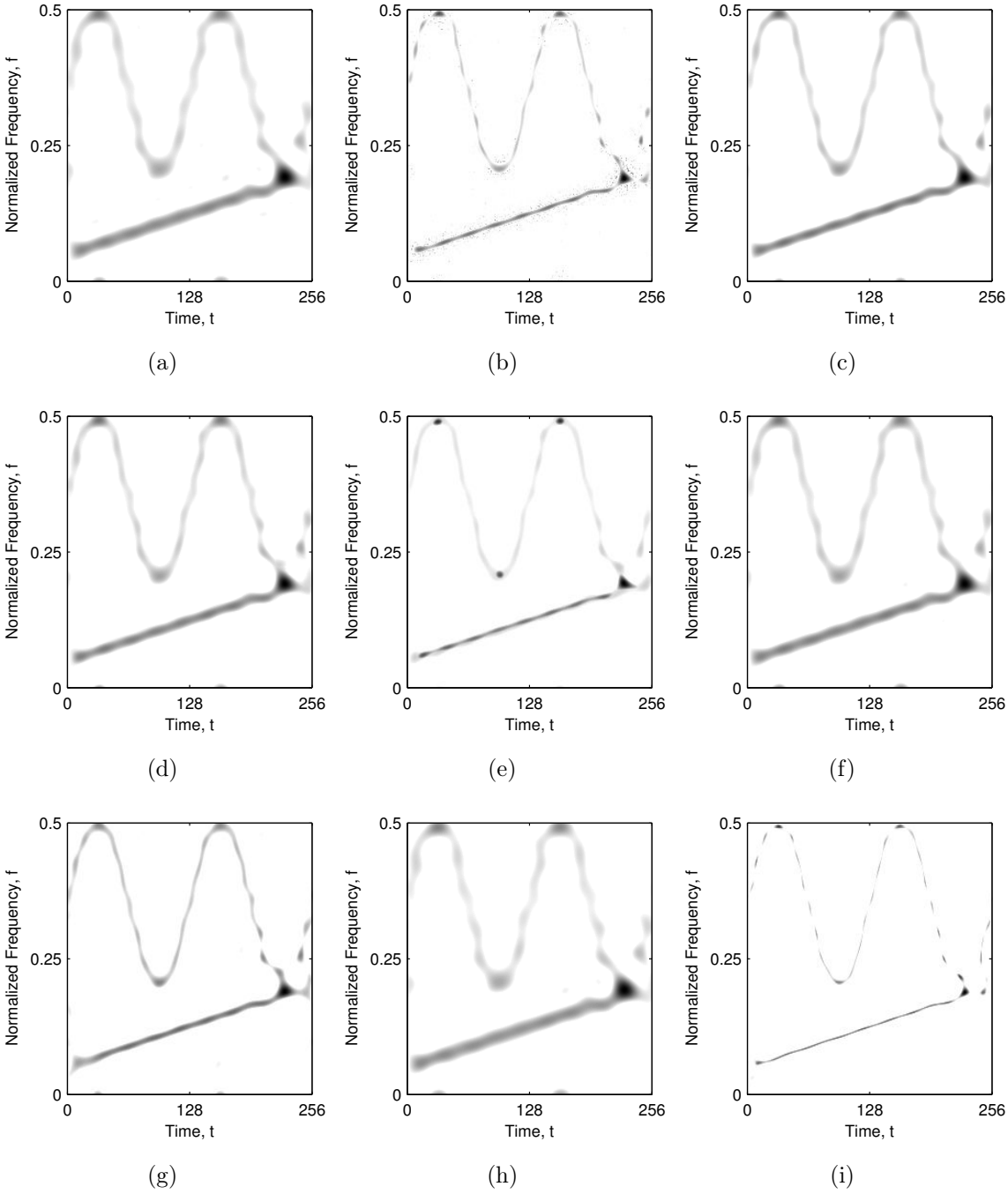


Figure 3.7: The reconstructed sparse TFDs based on the ℓ_1 norm minimization of the two component signal $z_{\mathrm{nonLFM}}(t)$ with: (a) IST, (b) TwIST, (c) FISTA, (d) SpaRSA, (e) FPC, (f) GPSR, (g) NESTA, (h) SALSA, (i) YALL1 algorithm.

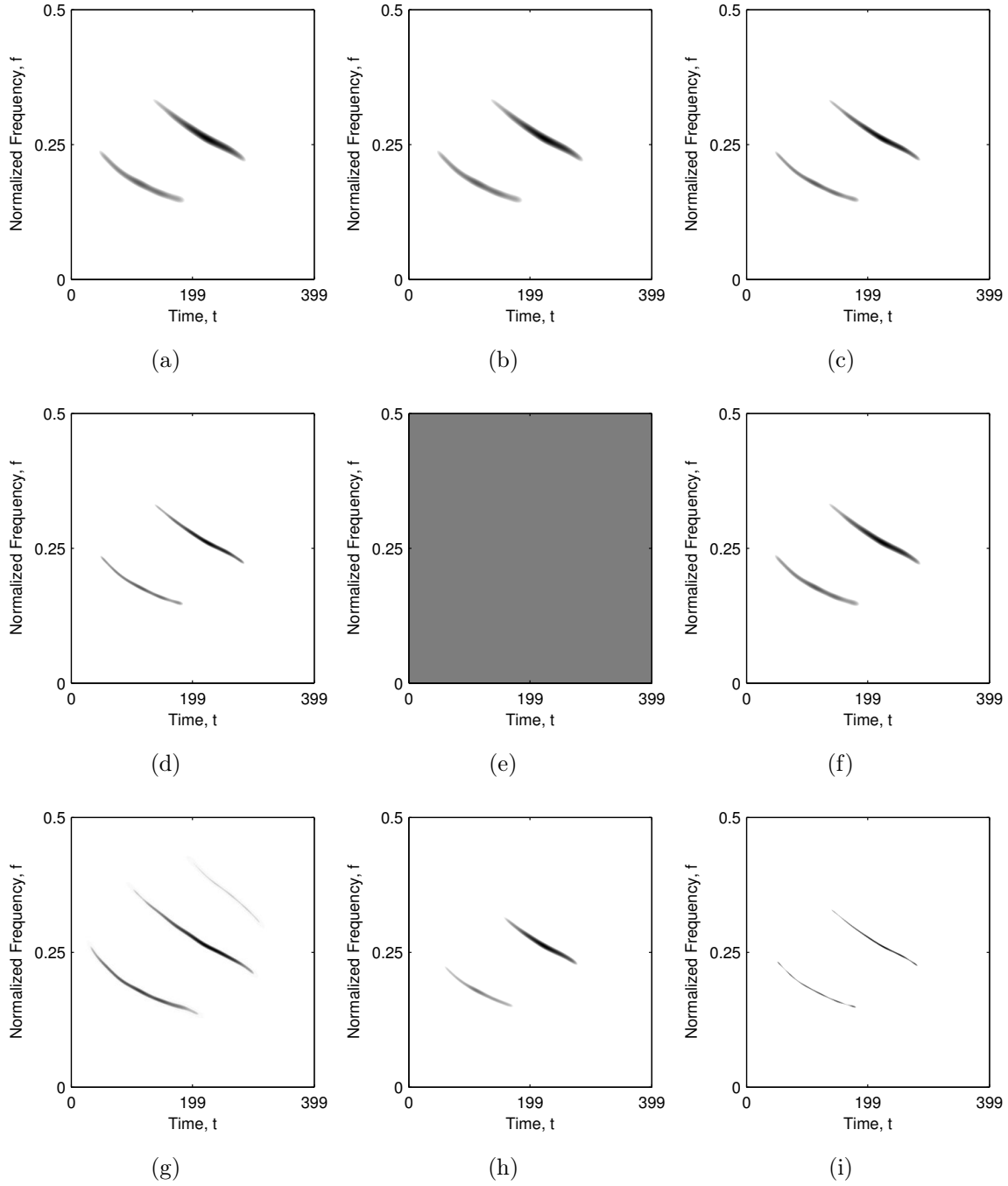


Figure 3.8: The reconstructed sparse TFDs based on the ℓ_1 norm minimization of the bat echolocation signal $z_{\mathrm{bat}}(t)$ with: (a) IST, (b) TwIST, (c) FISTA, (d) SpaRSA, (e) FPC, (f) GPSR, (g) NESTA, (h) SALSA, (i) YALL1 algorithm.

Table 3.2: Rényi entropies, $R_z^{\alpha_R}$, concentration measures, M_z^S, and the algorithm execution times of the reconstruction algorithms based on the ℓ_1 norm minimization. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{LFM}(t)$			$z_{nonLFM}(t)$			$z_{bat}(t)$		
	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_S = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_S = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_S = 2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IST	12.5535	0.1556	333.3987	12.4859	0.1513	209.5674	11.2552	0.0203	432.6409
TwIST	11.3956	0.0776	194.2069	11.2005	0.0648	159.2943	11.2552	0.0203	442.4628
FISTA	11.9123	0.0934	252.4976	12.0731	0.1063	167.2367	10.8487	0.0150	422.6172
SpaRSA	12.1135	0.1128	**160.8317**	12.1544	0.1195	**84.2935**	10.3820	0.0110	**278.7615**
FPC	11.5062	0.0869	894.3250	11.0434	0.0728	828.3579	–	–	–
GPSR	12.4003	0.1386	215.2057	12.3232	0.1311	151.6580	10.9132	0.0157	294.2261
NESTA	11.5276	0.1564	1584.2	11.8304	0.1997	1101.9	11.6021	0.2801	2608.2
SALSA	12.8112	0.2059	194.8357	12.4511	0.1733	195.5804	10.1860	0.0158	1304.6
YALL1	**9.9986**	**0.0227**	2027.3	**9.7727**	**0.0198**	2002.9	**9.0057**	**0.0041**	9339.8
ℓ_1-ls	11.8612	0.3577	4469.1	12.1695	0.4985	2390.5	14.4945	0.5674	2814.4

generalized across the tested algorithms. By visual inspection of the obtained results, it can be seen that the resulting TFDs are the best ones yet obtained, being better then the classically filtered TFDs (shown in Figs. 2.8 - 2.10), the RGK-TFD (shown in Fig. 2.11), the ℓ_2 norm based sparse TFDs (shown in Fig. 3.2), and the ℓ_0 norm based sparse TFDs (shown in Figs. 3.3 - 3.5). This is confirmed by very low values of the obtained concentration measures, which for some tested algorithms are close to the concentration measures of the ideal signal TFD. Note however, that the obtained algorithm execution times are significantly higher than the execution times of the algorithms based on the ℓ_0 norm minimization, shown in Table 3.1. This is because the ℓ_0 norm based sparse reconstruction algorithms are based on the hard-thresholding operator rather than the soft-thresholding operator, and the hard-thresholding operator tends to decrease the values of the TFD samples much faster, especially if implemented with the H_K operator.

3.4 Summary

In this chapter, the compressive sensing based approach for the TFD concentration enhancement has been discussed. The approach is based in the AF by compressively sensing the samples in such a way to include only the signal components, while discarding the interfering ones. The resolution loss, caused by the CS can be avoided by the reconstruction algorithms based on the sparsity constraint. Recently developed state-of-the-art sparse reconstruction algorithms based on the ℓ_0 norm, the ℓ_1 norm, and the ℓ_2 norm minimization have been modified and applied for the TFD concentration enhancement. The

conducted experiments on the previously defined signals $z_{\mathrm{LFM}}(t)$, $z_{\mathrm{nonLFM}}(t)$, and $z_{\mathrm{bat}}(t)$, have shown that the sparse TFDs obtained by the ℓ_1 norm minimization are superior to the classically filtered TFDs, the RGK-TFD, and the sparse TFDs obtained by the ℓ_0 norm minimization and the ℓ_2 norm minimization.

Chapter 4

Supporting Methods Needed in Adaptive Time-Frequency Compressive Sensing

In this chapter, the three adaptive methods have been discussed, used as supporting methods in the CS based TFD concentration enhancement, discussed later in Chapter 5.

The adaptive parallelogram 1/0 kernel is proposed, which is mathematically less demanding than the RGK, previously discussed in Section 2.4.2. Performance of the proposed kernel is tested on the previously defined synthetical and real-life signals, while the obtained TFDs are compared with the RGK-TFD performance.

The denoising methods developed by combining the local polynomial approximation (LPA) method with the rules derived from the intersection of confidence intervals (ICI) rule, namely the relative ICI (RICI) rule and the fast ICI (FICI) rule are discussed. The denoising capabilities of the LPA-ICI and the LPA-RICI methods are tested on the commonly used synthetical test signals, namely the *Blocks* signal and the *HeaviSine* signal. The FICI rule has been proposed and performance of the LPA-FICI method is compared with the performance of the LPA-ICI method and the LPA-RICI method.

The method which estimates the instantaneous number of signal components of the TFD is discussed. The method is based on the Rényi entropy counting property, and the method is tested on the previously defined synthetical (with the *a priori* known number of instantaneous signal components) and real-life signals.

4.1 Adaptive Parallelogram 1/0 Kernel

In this section, the new adaptive kernel design has been proposed, mathematically significantly less demanding then the RGK, previously discussed in Section 2.4.2, which is one

of the best performing adaptive TFD kernels. The proposed adaptive kernel; however, is unreliable for the signals with a nonlinear frequency modulated component. The kernel is build as a parallelogram centered around the AF origin, going through all the lag instances, with its height and slope defined by the kernel parameters N'_τ and N'_ν calculated from the signal AF geometry.

The kernel parameters N'_τ and N'_ν in the proposed method are calculated to be equal to the time and frequency distance between the two closest components in the TF domain, respectively, or equivalently in the AF domain, to the lag and doppler distance between the origin and the first pair of cross-terms. The proposed method is based on the AF domain approach, by searching zero doppler ($A_z(0,\tau)$) and zero lag ($A_z(\nu,0)$) slices for the first energy spike located away from the origin. The detected points on lag and doppler axes are in fact the points where the closest pair of cross-terms are located. The complete algorithm for finding N'_τ and N'_ν is defined in details in the sequel of this section, and is implemented in a MATLAB function `fun_adaptiveCSAF.m`, the code of which is given in Appendix B.

1. Since the AF domain is symmetrical, just one half of the lag and doppler AF zero slices are obtained:

$$S_\tau(\tau) = A_z(0, N_\tau/2 \to N_\tau), \tag{4.1a}$$
$$S_\nu(\nu) = A_z(N_\nu/2 \to N_\nu, 0), \tag{4.1b}$$

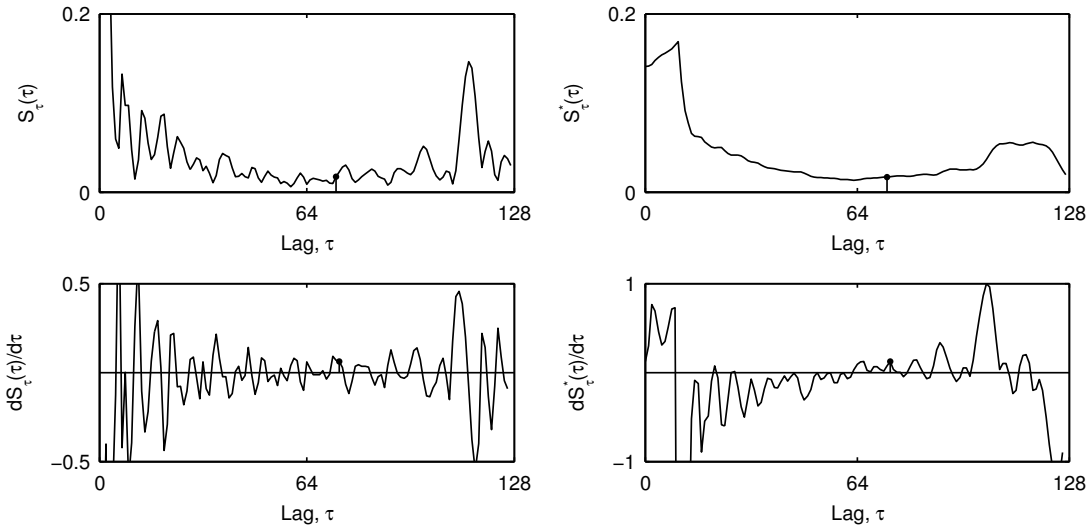
   where notation $N_1 \to N_2$ is used to define a subset of samples of the respective AF zero slice that are preserved for further processing, while N_τ and N_ν are the number of available lag instances and doppler bins, respectively.

2. In the next step, the derivatives of the zero-doppler and the zero-lag slices are calculated. The AF zero slices are highly oscillatory, as shown in Fig. 4.1(a) for the AF zero-doppler slice of the signal $z_{\mathrm{LFM}}(t)$, and thus their derivatives will also oscillate around the respectful axis, meaning that the cross-terms location information would be hard to extract from them. This problem is solved by filtering the AF zero slices, that is:

$$S^*_\tau(\tau) = \mathrm{MA}\{S_\tau(\tau)\}, \tag{4.2a}$$
$$S^*_\nu(\nu) = \mathrm{MA}\{S_\nu(\nu)\}, \tag{4.2b}$$

   where the operator $\mathrm{MA}\{\cdot\}$ denotes the moving average filter, defined as:

$$\mathrm{MA}\{x(t)\} = \frac{1}{\Delta t} \int_{t-\Delta t/2}^{t+\Delta t/2} x(t)dt. \tag{4.3}$$



(a) (b)

Figure 4.1: Zero-doppler slice of the signal $z_{\mathrm{LFM}}(t)$ with marked N'_τ: (a) the unfiltered slice, $S_\tau(\tau)$ and its derivative, $dS_\tau(\tau)/d\tau$, (b) the filtered slice, $S^*_\tau(\tau)$, and its derivative, $dS^*_\tau(\tau)/d\tau$.

From the extensive simulations, it has been concluded that the moving average filter window needs to be long enough to smoothen up the respectful slice, revealing its global trend-line, as shown in Fig. 4.1(b). The conducted simulations involving various test signals have shown that $N_\tau/12$ and $N_\nu/12$ are good choice for the moving average filters window sizes.

3. Finally, the parameters N'_τ and N'_ν are calculated by searching $dS^*_\tau(\tau)/d\tau$ and $dS^*_\nu(\nu)/d\nu$ for a location of local maximum after the first crossing from negative to positive values. By doing so, auto-terms located at beginning of the slices can settle down, ensuring that detected peak is caused by the cross-terms. To make the method even more robust, two additional conditions have been included: the next sample of $dS^*_\tau(\tau)/d\tau$ and $dS^*_\nu(\nu)/d\nu$ must be positive as well, and it has to be above a preset threshold value, ϵ_{APK} (in the examples to follow, the threshold value is set to $\epsilon_{\mathrm{APK}} = 20\%$ of the slices maximum value).

The here-proposed method can detect the cross-terms location, even in the case when the cross-terms do not intersect with the AF axes, because from the AF volume distribution relationships [47]:

$$\int_{-\infty}^{\infty} |A_z(\nu,\tau)|^2 d\tau = \int_{-\infty}^{\infty} |A_z(0,\tau)|^2 e^{-j2\pi\nu\tau} d\tau, \tag{4.4a}$$

$$\int_{-\infty}^{\infty} |A_z(\nu,\tau)|^2 d\nu = \int_{-\infty}^{\infty} |A_z(\nu,0)|^2 e^{j2\pi\nu\tau} d\nu, \tag{4.4b}$$

it can be concluded that the cross-terms are, in fact, detectable on the AF zero slices, even when they are not intersecting it.

Another way to understand this is to look at the AF as a two-dimensional auto-correlation function. This approach can be confirmed by combing (2.25) and (2.27), and substituting $\nu = 0$, in order to obtain the AF zero doppler slice, $A_z(0, \tau)$:

$$A_z(0,\tau) = \int_{-\infty}^{\infty} z\left(t + \frac{\tau}{2}\right) z^*\left(t - \frac{\tau}{2}\right) d\tau, \tag{4.5}$$

which gives the well known expression for the one-dimensional auto-correlation function. In the similar way, other AF slices can be interpreted as cross-correlation between the signal $z(t)$ and its frequency shifted copies $z(t)e^{-j2\pi\nu t}$. Furthermore, it is well known that the auto-correlation, along with the cross-correlation function exhibit cross-terms between the components. This fact lead us to the same conclusion as before, that the cross-terms do not need to intersect with the AF axis, in order to make their location detectable by the proposed algorithm.

With the parameters N'_τ and N'_ν selected in this way, the adaptive kernel is build as a parallelogram, centered around the AF origin, and going through all the lag instances, that is:

$$g(\nu,\tau) = \begin{cases} 1, & \left|\dfrac{k_{\text{APK}}}{N'_\nu}\nu - \dfrac{2}{N'_\tau}\tau\right| \leq 1, \\ 0, & \text{otherwise}, \end{cases} \tag{4.6}$$

where the kernel parameter k_{APK} is selected as either -1 or 1, depending on the auto-term direction. The here-proposed kernel is denoted as the adaptive parallelogram 1/0 kernel (APK).

Example 4.1. In this example, the APK and the associated TFDs of the previously defined signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$ have been calculated and the results are shown in Fig. 4.2.

For the signal $z_{\text{LFM}}(t)$, as it can be seen from Fig. 4.2(a), the APK perfectly captured all the auto-terms, while avoiding the cross-terms. When compared to the RGK performance, shown in Fig. 2.11, it can be seen that the APK performance is very similar to performance of the RGK. The difference in the concentration measures is caused by the additional RGK-TFD smoothing, which can be performed in the same way for the APK-TFD. However, for the signal $z_{\text{nonLFM}}(t)$, the APK performance is significantly worse when compared to the RGK performance. This was expected, since the signal in question contains a highly nonlinear frequency modulated component, which resulted that APK did not manage to properly capture the auto-terms, as it can be seen in Fig. 4.2(b). On the other hand, for the signal $z_{\text{bat}}(t)$, the APK runs very competitively with the RGK,

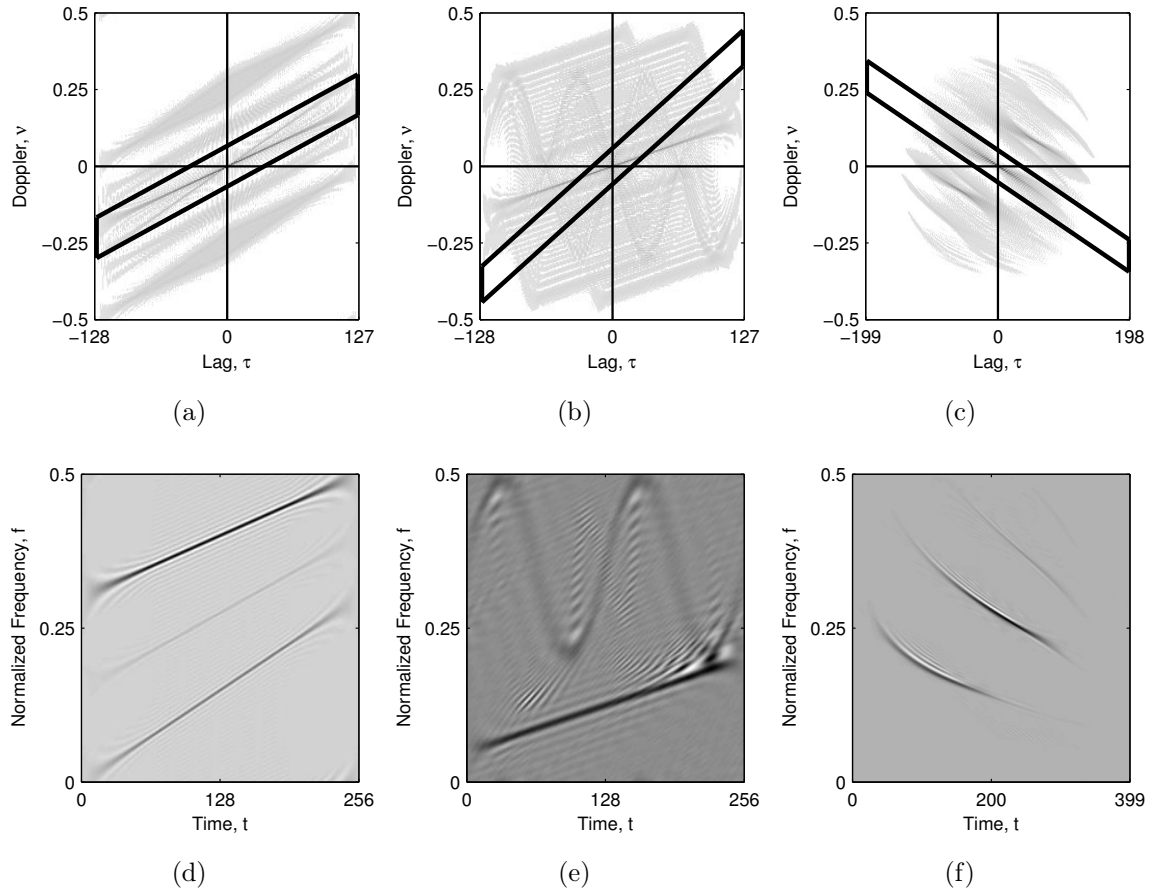


Figure 4.2: The performance of the adaptive parallelogram 1/0 kernel: (a)-(c) AF with marked kernel area of the three LFM component signal, $z_{\mathrm{LFM}}(t)$ ($N'_\tau = 73$ and $N'_\nu = 17$), the two component signal, $z_{\mathrm{nonLFM}}(t)$ ($N'_\tau = 39$ and $N'_\nu = 15$), and the bat echolocation signal, $z_{\mathrm{bat}}(t)$ ($N'_\tau = 71$ and $N'_\nu = 21$), respectively; (d)-(f) resulting TFDs of the three LFM component signal, $z_{\mathrm{LFM}}(t)$ ($M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 1.0736$, and $R_z^{\alpha_{\mathrm{R}}}\big|_{\alpha_{\mathrm{R}}=3} = 10.4168$), the two component non-LFM signal, $z_{\mathrm{nonLFM}}(t)$ ($M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 1.7917$, and $R_z^{\alpha_{\mathrm{R}}}\big|_{\alpha_{\mathrm{R}}=3} = 11.2063$), and the bat echolocation signal, $z_{\mathrm{bat}}(t)$ ($M_z^{\mathrm{S}}\big|_{p_{\mathrm{S}}=2} = 0.5906$, and $R_z^{\alpha_{\mathrm{R}}}\big|_{\alpha_{\mathrm{R}}=3} = 10.8930$), respectively.

even without the additional smoothing. The RGK-TFD has a slightly better concentration measure, however, the APK-TFD did not filtered-out the weakest signal component. The reason behind the APK good performance, for the given signal with the nonlinear frequency modulated components, is that the nonlinearity in the signal components frequency modulation has been compensated with the difference between the auto-terms slopes, as it can be seen in Fig. 4.2(c).

The main problem of the here-proposed kernel design, as illustrated in Example 4.1, is when the observed signal has a highly nonlinear frequency modulated component. The here-proposed algorithm determines the auto-terms location based on the cross-terms location, assuming a linear behaviour of the auto-terms. However, when this is not a case, the cross-terms and the auto-terms location are not necessarily connected in the same manner (depending on type of the nonlinearity in the signal component phase), resulting in the unreliable kernel. This is why in Section 5.1 a different way to construct

the more reliable TFD kernel has been proposed using the same parameters N'_τ and N'_ν, but in combination with the sparse signal reconstruction algorithms, discussed in Section 3.3, preventing the auto-terms resolution loss.

4.2 Denoising Methods Based on the Intersection of Confidence Intervals Rule

4.2.1 Motivation

The work presented in this Section is based on the work presented in the [79]. Acquired real-life signals are often noisy due to the imperfection in their acquisition, transmission, and/or processing procedures. In order to efficiently extract desired information from such signals, it is essential to remove or suppress the noise that corrupts useful data. In general, denoising methods can be divided into two broad groups: parametric and non-parametric estimators [43]. Parametric estimators require *a priori* knowledge about signal and noise probability density functions (PDFs). When this assumptions are correct, parametric estimators will produce more precise estimates than non-parametric estimators. However, in most practical applications, such *a priori* knowledge is not accessible, thus non-parametric estimators have to be used. Most crucial parameter in non-parametric estimators is the filter support size. The optimal size is defined by the compromise between estimation bias and variance. Small filter support size will result in estimate with large variance, hence resulting in undersmoothing, while large filter support size will cause uncontrollable estimation bias, resulting in oversmoothing [43].

The non-parametric methods which are based on estimation of bias and variance are called "plug-in" methods. Those methods, however, require high-order derivatives of the estimate, hence they are more complex to implement and computationally very demanding [43]. Alternatives to "plug-in" methods are the methods based on the quality-of-fit statistics which do not require the knowledge of the estimation bias. One such quality-of-fit signal denoising method is the LPA-ICI method [34, 41], requiring only noise standard deviation, which is easily obtained in the case of additive white Gaussian noise (AWGN). The LPA-ICI method combines LPA and ICI (originally developed in [34], and improved in [41]) methods such that the ICI algorithm is used for the filter support size selection, which is then complemented by the LPA method, used as the filter design tool.

4.2.2 The LPA-ICI method

A noise corrupted signal $x(t)$ is composed of a noise-free signal $s(t)$ and AWGN, $\epsilon(t)$, that is: $x(t) = s(t) + \epsilon(t)$. The goal of the signal denoising is to estimate $\hat{s}(t, w)$ such that the

LPA absolute estimation error:

$$|e(t,w)| = |s(t) - \hat{s}(t,w)|, \tag{4.7}$$

is reduced as much as possible, where $w(t)$ is an adaptive filter support size. The filter support size plays a key role in the performance of the LPA estimators by controlling the trade-off between the variance and the bias. Thus, the denoising problem reduces to finding such a filter support length, $w^+(t)$, which results in the optimal bias-variance trade-off [43].

The point-wise mean square risk, $r(t,w)$, is expressed as [42]:

$$r(t,w) = E\{e^2(t,w)\} = \text{std}^2\{\hat{s}(t,w)\} + |\text{bias}\{e(t,w)\}|^2, \tag{4.8}$$

where $\text{std}\{\hat{s}(t,w)\}$ is the standard deviation of the estimate, and $\text{bias}\{e(t,w)\}$ is the bias of the estimation error. The minimization of the mean square risk, $r^+(t,w)$, over the window length, $w(t)$, results in an ideal filter support size $w^+(t)$ [42]:

$$r^+(t,w) = \min(r(t,w)) = \text{std}^2\{\hat{s}(t,w^+)\}(1+\eta^2_{\text{ICI}}), \tag{4.9}$$

where η_{ICI} at $w(t) = w^+(t)$ can be expressed as:

$$\eta_{\text{ICI}} = \frac{\text{bias}\{e(t,w^+)\}}{\text{std}\{\hat{s}(t,w^+)\}}. \tag{4.10}$$

In the case of LPA estimates, the following inequality holds [41]:

$$|e(t,w)| \leq \text{bias}\{e(t,w)\} + |\zeta(t,w)|, \tag{4.11}$$

where $\zeta(t,w) \sim \mathcal{N}(0, \text{std}^2\{\hat{s}(t,w)\})$ is the zero mean Gaussian random error with standard deviation $\text{std}\{\hat{s}(t,w)\}$. With the probability $p = 1-\alpha$, following inequality holds true: $|\zeta(t,w)| \leq \chi_{1-\alpha/2} \cdot \text{std}\{\hat{s}(t,w)\}$ [41], thus with the same probability inequality (4.11) leads to:

$$|e(t,w)| \leq \text{bias}\{e(t,w)\} + \chi_{1-\alpha/2} \cdot \text{std}\{\hat{s}(t,w)\}, \tag{4.12}$$

where $\chi_{1-\alpha/2}$ is $(1-\alpha/2)$th quantile of the standard Gaussian distribution.

The ICI rule introduces a finite set of filter support sizes $\{w_1 < w_2 < \cdots < w_L\}$ giving the proper filter support size as the one which provides adequate compromise between the estimation error bias, $\text{bias}\{e(t,w)\}$, and the random error, $\zeta(t,w)$. Thus, inequality

(4.12) can be redefined:

$$|e(t,w)| \leq (\eta_{\text{ICI}} + \chi_{1-\alpha/2}) \cdot \text{std}\{\hat{s}(t,w)\}, \tag{4.13a}$$

$$|s(t) - \hat{s}(t,w)| \leq \Gamma_{\text{ICI}} \cdot \text{std}\{\hat{s}(t,w)\}, \tag{4.13b}$$

where parameter $\Gamma_{\text{ICI}} = (\eta_{\text{ICI}} + \chi_{1-\alpha/2})$ is the ICI threshold value. With the same probability, p, inequality (4.14) can be expanded into:

$$\hat{s}(t,w) - \Gamma_{\text{ICI}} \cdot \text{std}\{\hat{s}(t,w)\} \leq s(t) \leq \hat{s}(t,w) + \Gamma_{\text{ICI}} \cdot \text{std}\{\hat{s}(t,w)\}, \tag{4.14}$$

which leads to the introduction of confidence intervals, $\hat{s}(t,w) \in D(t)$, with upper and lower boundaries respectively being defined as [41]:

$$D_{\text{u}}(t_0 + \Delta t) = \hat{s}(t_0 + \Delta t, w) + \Gamma_{\text{ICI}} \cdot \text{std}\{\hat{s}(t_0 + \Delta t, w)\}, \tag{4.15a}$$

$$D_{\text{l}}(t_0 + \Delta t) = \hat{s}(t_0 + \Delta t, w) - \Gamma_{\text{ICI}} \cdot \text{std}\{\hat{s}(t_0 + \Delta t, w)\}, \tag{4.15b}$$

with $t = t_0 + \Delta t(t_0)$, where t_0 is the considered signal sample, $\Delta t(t_0)$ is an element of $[0, N_t - t_0]$ for the forward calculation, and an element of $[0, -t_0]$ for the backwards calculation of the confidence intervals intersection, and $\hat{s}(t_0 + \Delta t)$ is calculated using the LPA method (for zero order LPA, its value is calculated as the average value of the samples in the vicinity of the considered signal sample, detected using the ICI algorithm).

The ICI algorithm results in the number of samples $\Delta t^+(t_0)$ which is then used for the adaptive filter support size selection, where $\Delta t^+(t_0)$ is the largest $\Delta t(t_0)$ satisfying the following condition [41]:

$$D_{\text{u}^+_{\text{min}}}(t_0 + \Delta t) \geq D_{\text{l}^+_{\text{max}}}(t_0 + \Delta t), \tag{4.16}$$

where $D_{\text{u}^+_{\text{min}}}(t_0 + \Delta t)$ and $D_{\text{l}^+_{\text{max}}}(t_0 + \Delta t)$ are calculated as:

$$D_{\text{u}^+_{\text{min}}}(t_0 + \Delta t) = \min(D_{\text{u}}(t_0), \ldots, D_{\text{u}}(t_0 + \Delta t)), \tag{4.17a}$$

$$D_{\text{l}^+_{\text{max}}}(t_0 + \Delta t) = \max(D_{\text{l}}(t_0), \ldots, D_{\text{l}}(t_0 + \Delta t)). \tag{4.17b}$$

The most appropriate size of the support window for the considered signal sample, $w^+(t_0)$, is calculated as:

$$w^+(t_0) = \Delta t^+_{\text{f}}(t_0) + \Delta t^+_{\text{b}}(t_0) + 1, \tag{4.18}$$

where $\Delta t^+_{\text{f}}(t_0)$ and $\Delta t^+_{\text{b}}(t_0)$ are the number of samples obtained from the forward and backward confidence intervals intersection calculation, respectively. The flowchart of the

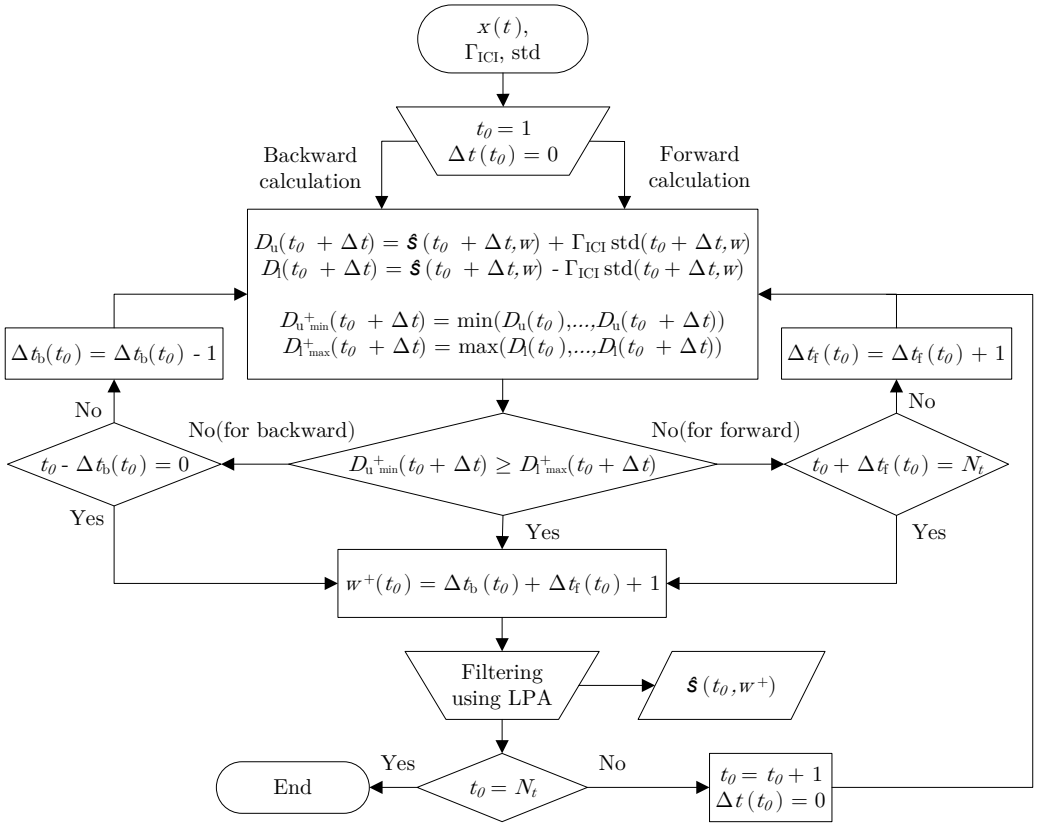


Figure 4.3: Flowchart of the ICI algorithm.

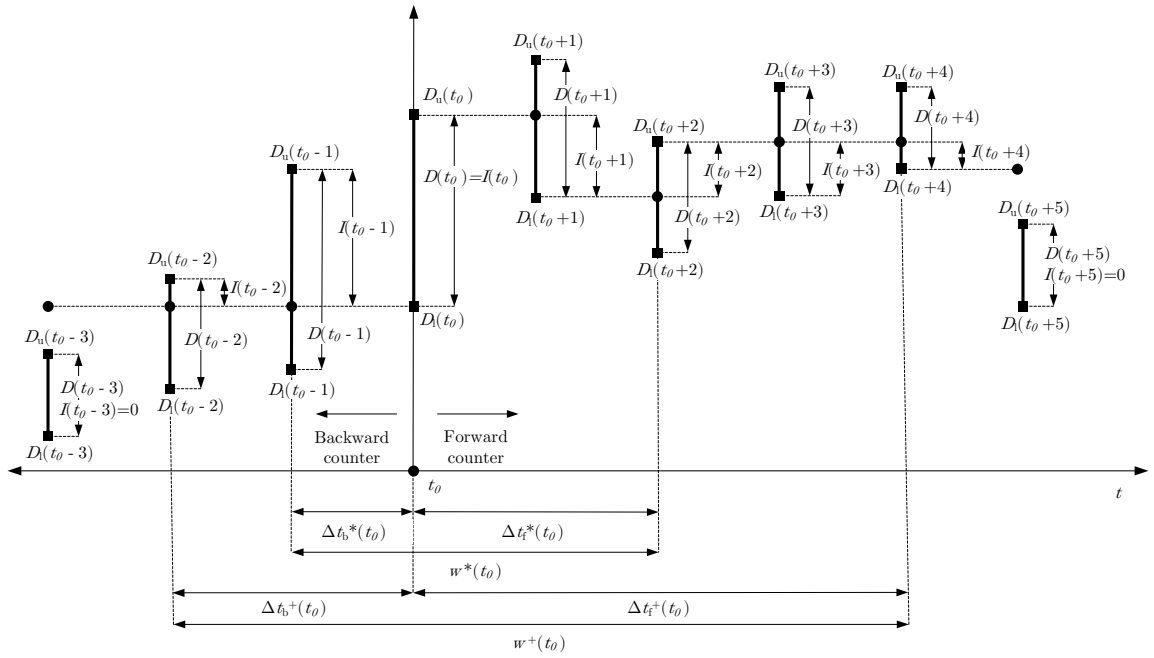


Figure 4.4: Asymmetrical filter support size selection. $\Delta t_{\mathrm{b}}^{+}(t_0)$, $\Delta t_{\mathrm{f}}^{+}(t_0)$, and $w^{+}(t_0)$ are determined using the original ICI algorithm, while $\Delta t_{\mathrm{b}}^{*}(t_0)$, $\Delta t_{\mathrm{f}}^{*}(t_0)$, and $w^{*}(t_0)$ are obtained after introducing (4.21) as an additional criterion.

ICI algorithm described in (4.15)-(4.18) is given in the Fig. 4.3.

An example of confidence intervals intersection, with the confidence intervals lengths, $D(t_0+\Delta t)$, and the amount of confidence intervals overlaps, $I(t_0+\Delta t)$, is illustrated in Fig. 4.4. In the forward ICI calculation, the intervals are intersecting for $\Delta t_{\mathrm{f}}(t_0) = 1, \ldots, 4$, however there are no intersections for $\Delta t_{\mathrm{f}}(t_0) = 5$, thus the optimal forward window size is selected as $\Delta t_{\mathrm{f}}^{+}(t_0) = 4$. In a similar way, in the backward ICI calculation, the intervals are intersecting for $\Delta t_{\mathrm{b}}(t_0) = 1, 2$, however there are no intersections for $\Delta t_{\mathrm{b}}(t_0) = 3$, thus the optimal forward window size is selected as $\Delta t_{\mathrm{b}}^{+}(t_0) = 2$, resulting in the optimal asymmetrical window of size $w^{+}(t_0) = 4 + 2 + 1 = 7$.

Example 4.2. In this example the LPA-ICI algorithm denoising performance has been tested on the standard test signals *Blocks* and *HeaviSine*[1] [41, 46], shown in Figs. 4.5(a) and 4.5(b) with their lengths varying from $N_t = 256$ to $N_t = 2048$ samples. The *Blocks* signal is known to be a model of acoustic impedance of a layered medium in geophysics, as well as one-dimensional profile of images in some image processing problems, while the *HeaviSine* signal is a sinusoidal signal with jumps (obtained by combining the sinusoid and a piecewise constant component). Furthermore, the test signals have been corrupted with the AWGN with the signal-to-noise ratio SNR = 10 dB (noisy signals are shown in Figs. 4.5(c) and 4.5(d)), and the results have been averaged over $M = 100$ Monte Carlo simulations. The LPA-ICI algorithm threshold parameters have been fixed at $\Gamma_{\mathrm{ICI}} = 3$, and the algorithms denoising quality has been measured with the mean square error (MSE), the root of mean square error (RMSE), the mean absolute error (MAE), and the maximum absolute difference (MAX), calculated as:

$$\mathrm{MSE} = \frac{1}{MN_t} \sum_{m=1}^{M} \sum_{t=1}^{N_t} \left(s(t) - \hat{s}^{[m]}(t, w) \right)^2, \tag{4.19a}$$

$$\mathrm{RMSE} = \frac{1}{M} \sum_{m=1}^{M} \left(\frac{1}{N_t} \sum_{t=1}^{N_t} \left(s(t) - \hat{s}^{[m]}(t, w) \right)^2 \right)^{\frac{1}{2}}, \tag{4.19b}$$

$$\mathrm{MAE} = \frac{1}{MN_t} \sum_{m=1}^{M} \sum_{t=1}^{N_t} \left| s(t) - \hat{s}^{[m]}(t, w) \right|, \tag{4.19c}$$

$$\mathrm{MAX} = \frac{1}{M} \sum_{m=1}^{M} \max \left(\left| s(t) - \hat{s}^{[m]}(t, w) \right| \right), \tag{4.19d}$$

respectively, where $\hat{s}^{[m]}(t, w)$ is the estimated signal value in the m-th simulation. The obtained denoised signals are shown in Figs. 4.5(e) and 4.5(f), and are summarized in Table 4.1.

The algorithm execution times are obtained on the PC with the Intel Core i7-4770

[1] Part of the Wavelab toolbox, available at: `http://www-stat.stanford.edu/~wavelab/Wavelab_850/WAVELAB850.ZIP`

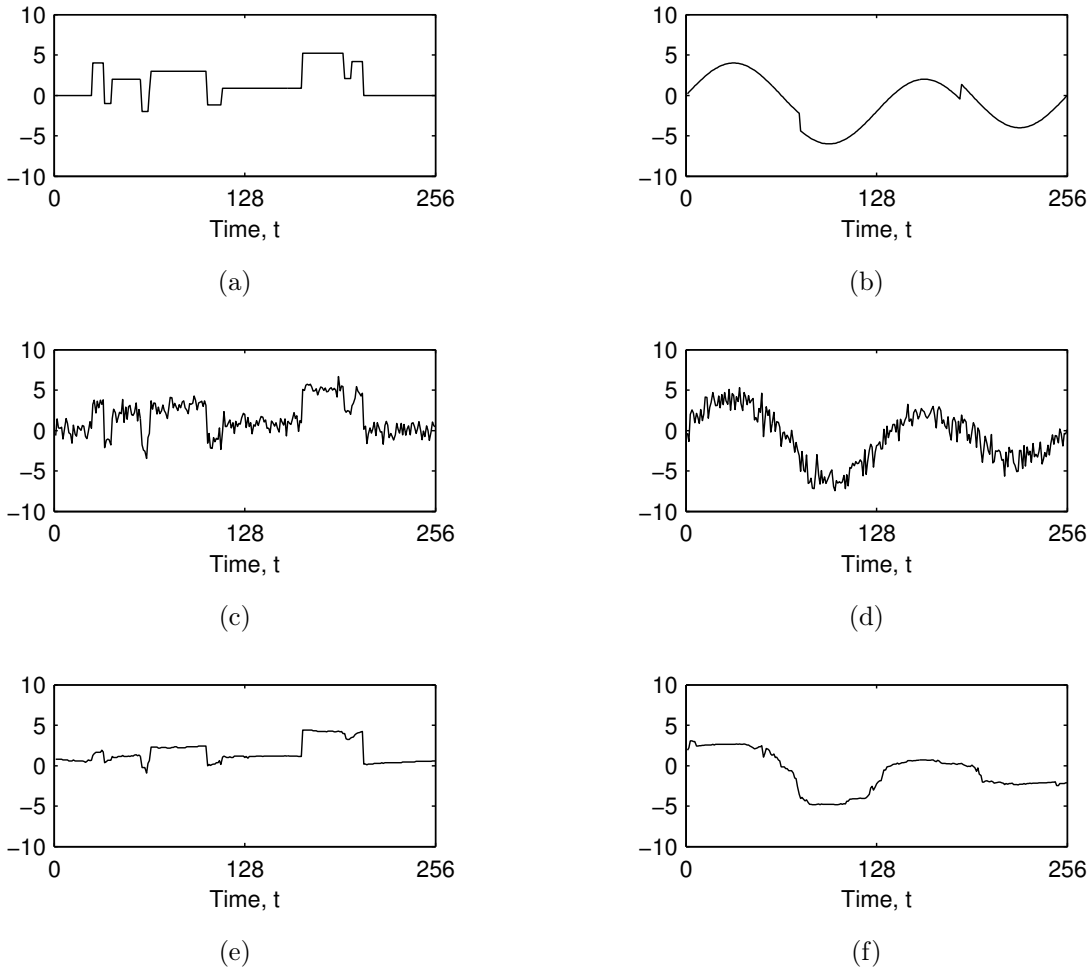


Figure 4.5: Used test signals with $N_t = 256$ samples: (a) *Blocks* signal, (b) *HeaviSine* signal, (c) noisy *Blocks* signal (SNR = 10 dB), (d) noisy *HeaviSine* signal (SNR = 10 dB), (e) denoised *Blocks* signal with the LPA-ICI algorithm ($\Gamma_{\text{ICI}} = 3$), (f) denoised *HeaviSine* signal with the LPA-ICI algorithm ($\Gamma_{\text{ICI}} = 3$).

Table 4.1: Signal denoising results of the LPA-ICI algorithm with $\Gamma_{\text{ICI}} = 3$.

	$N_t = 256$		$N_t = 512$		$N_t = 1024$		$N_t = 2048$	
	Blocks	*HeaviSine*	*Blocks*	*HeaviSine*	*Blocks*	*HeaviSine*	*Blocks*	*HeaviSine*
MSE	0.8476	0.9462	0.5056	0.5489	0.3018	0.3085	0.1530	0.1775
RMSE	0.9185	0.9713	0.7101	0.7396	0.5487	0.5545	0.3907	0.4207
MAE	0.7530	0.8335	0.5599	0.6271	0.4298	0.4662	0.3057	0.3471
MAX	2.8508	2.2691	2.4741	1.8541	2.3438	1.5997	2.2801	1.3451
t [ms]	0.0571	0.0699	0.1878	0.2207	0.6689	0.7014	2.5742	2.2560

@ 3.40 Ghz processor and 16 GB of RAM, and should be taken only in comparison among themselves. Due to the nature of the LPA-ICI algorithm, its execution time grows quadratically with the signal length, i.e. the LPA-ICI algorithm execution time is increased 45 times, when signal length is increased from $N_t = 256$ samples to $N_t = 2048$ samples. Furthermore, execution time of the LPA-ICI algorithm is also dependent of signal type, which is obvious from the differences in algorithm execution times for *Blocks* and *HeaviSine* signals.

4.2.3 The LPA-RICI method

The original ICI algorithm is not only time consuming, but its efficiency is highly dependant on the value of the parameter Γ_{ICI} [41, 46]. For large Γ_{ICI} values, the ICI algorithm results in oversized filter supports, resulting in the signal oversmoothing. On the other hand, too small Γ_{ICI} values cause undersized filter supports and signal undersmoothing. This problem can be solved by finding a proper value of parameter Γ_{ICI} using the cross-validation or the variable Γ_{ICI} selection method [41]. Those methods, however, require additional reruns of the ICI algorithm, thus making it even more time consuming.

In order to solve the problem of finding a proper Γ_{ICI} value without significantly increasing the execution time, a new parameter has been introduced based on the overlap lengths between two consecutive confidence intervals [46]. The relative amount of confidence intervals overlapping is calculated as:

$$\begin{aligned} R(t_0+\Delta t) = \frac{I(t_0+\Delta t)}{D(t_0+\Delta t)} &= \frac{D_{\text{u}^+_{\min}}(t_0+\Delta t) - D_{\text{l}^+_{\max}}(t_0+\Delta t)}{D_{\text{u}}(t_0+\Delta t) - D_{\text{l}}(t_0+\Delta t)} \\ &= \frac{D_{\text{u}^+_{\min}}(t_0+\Delta t) - D_{\text{l}^+_{\max}}(t_0+\Delta t)}{2\Gamma_{\text{ICI}} \cdot \text{std}\{\hat{s}(t_0+\Delta t, w)\}}, \end{aligned} \tag{4.20}$$

where $D(t_0+\Delta t)$ and $I(t_0+\Delta t)$ are the length of confidence interval and the overlapping length with previous confidence intervals, respectively (as shown in Fig. 4.4). The new adaptive filter support size $w^*(t_0) = \Delta t^*(t_0)$ is then determined as the largest $\Delta t(t_0)$ satisfying the following condition, leading to the RICI rule:

$$R(t_0+\Delta t) \geq R_{\text{c}}, \tag{4.21}$$

where R_{c} is the preset threshold value. Note that setting $R_{\text{c}} = 0$ reduces the algorithm to the original ICI algorithm.

Example 4.3. In this example the LPA-RICI algorithm denoising performance has been tested, with identical simulation set-up as in Example 4.2, making obtained results mutually comparable. The LPA-RICI algorithm parameters have been set as: $\Gamma_{\text{ICI}} = 3$ and $R_c = 0.8$. The obtained denoised signals are shown in Fig. 4.6, while the denoising re-

Table 4.2: Signal denoising results of the LPA-RICI algorithm with $\Gamma_{\text{ICI}} = 3$ and $R_c = 0.8$.

	$N_t = 256$		$N_t = 512$		$N_t = 1024$		$N_t = 2048$	
	Blocks	*HeaviSine*	*Blocks*	*HeaviSine*	*Blocks*	*HeaviSine*	*Blocks*	*HeaviSine*
MSE	0.1015	0.3180	0.0643	0.2136	0.0456	0.1520	0.0347	0.1098
RMSE	0.3144	0.5621	0.2508	0.4613	0.2123	0.3891	0.1855	0.3310
MAE	0.1965	0.4462	0.1481	0.3589	0.1203	0.2938	0.0948	0.2427
MAX	2.1032	2.1044	2.3009	2.3820	2.5059	2.8385	2.6845	3.2396
t [ms]	0.0358	0.0334	0.1209	0.0943	0.4068	0.2706	1.3568	0.7803

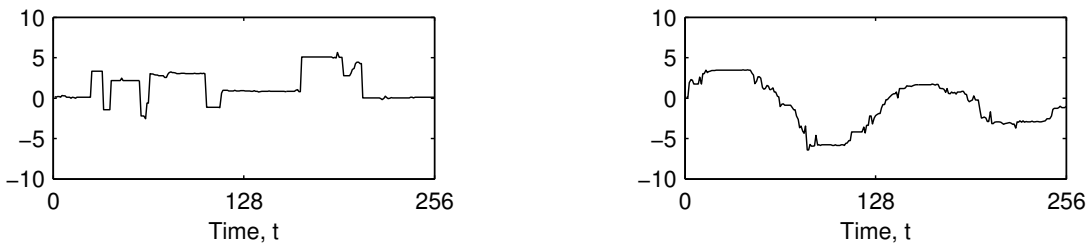
sults are summarized in Table 4.2. The LPA-RICI algorithm execution times are slightly smaller than the execution times of the LPA-ICI algorithm, however, the LPA-RICI algorithm suffers from the same problems as the LPA-ICI algorithm: non-linear increase of the algorithm execution time as signal length increases, and algorithm execution time is dependant of the signal type. When compared to the LPA-ICI denoising quality, the LPA-RICI denoising quality is significantly better across all of the calculated indices, except for the MAX value, which is in most of the performed simulations higher for the LPA-RICI algorithm.

4.2.4 The LPA-FICI method

As discussed in Section 4.2.2, the ICI algorithm at the signal sample t_0 calculates the following number of confidence intervals and their intersections:

$$N_{\text{CI}}^{\text{ICI}}(t_0) = \Delta t_{\text{f}}^*(t_0) + \Delta t_{\text{b}}^*(t_0) + 3 \ \ \in [3, N_t], \tag{4.22}$$

which makes $N_{\text{CI}}^{\text{ICI}}(t_0) \sim \mathcal{O}(N_t)$. This means that the ICI algorithm calculates minimum $N_{\text{CI}}^{\text{ICI}}(t_0) = 3$ confidence intervals (in the case when there is no intersection between them, resulting in $\Delta t_{\text{f}}^*(t_0) = \Delta t_{\text{b}}^*(t_0) = 0$), however the maximum number of confidence intervals calculation is $N_{\text{CI}}^{\text{ICI}}(t_0) = N_t$ (in the case where confidence intervals of all signal samples are intersecting). The ICI algorithm repeats the described procedure for each signal sample, that is: $[t_0]^{[n+1]} = [t_0]^{[n]} + 1$, and (4.22) can be expanded to take into account the total



(a) (b)

Figure 4.6: Denoised test signals with the LPA-RICI algorithm ($\Gamma_{\text{ICI}} = 3$, $R_c = 0.8$): (a) *Blocks* signal, (b) *HeaviSine* signal.

number of calculated confidence intervals and their intersections for the entire signal:

$$N_{\mathrm{CI}}^{\mathrm{ICI}} = \sum_{t_0=0}^{N_t} (\Delta t_{\mathrm{f}}^*(t_0) + \Delta t_{\mathrm{b}}^*(t_0) + 3) \quad \in [3N_t, N_t^2], \tag{4.23}$$

which makes $N_{\mathrm{CI}}^{\mathrm{ICI}} \sim \mathcal{O}(N_t^2)$, as demonstrated in Examples 4.2 and 4.3. This significantly slows down the ICI algorithm, especially for large chunks of data, and making it not applicable in real-time signal processing since it requires future signal samples.

The motivation behind the here-proposed FICI algorithm is to obtain the filter support size in groups, by dividing the signal in regions of varying size. The calculation of the filter support in such a way eliminates the need for calculating $\Delta t_{\mathrm{b}}^*(t_0)$, thus the FICI algorithm at the signal sample t_0 calculates the following number of confidence intervals and their intersections:

$$N_{\mathrm{CI}}^{\mathrm{FICI}}(t_0) = \Delta t_{\mathrm{f}}^*(t_0) + 2 \quad \in [2, N_t]. \tag{4.24}$$

However, the FICI algorithm skips the filter support size calculation for samples in the current group assuming they have the same support size, that is:

$$w(t_0)^* = w(t_0 + 1)^* = \cdots = w(t_0 + \Delta t_{\mathrm{f}}^*(t_0))^*, \tag{4.25}$$

and jumps to the first sample outside the current group, that is $[t_0]^{[n+1]} = [t_0]^{[n]} + \Delta t_{\mathrm{f}}^*(t_0) + 1$. Thus, the total number of calculated confidence intervals and their intersections for the entire signal stays the same, i.e. one per signal sample:

$$N_{\mathrm{CI}}^{\mathrm{FICI}} = N_t, \tag{4.26}$$

making the algorithm appealing in many applications, since its execution time can be easily and accurately approximated on any system based on the execution time of the test signal with smaller number of samples. By comparing (4.26) with the (4.23), it can be clearly seen that the LPA-FICI algorithm reduces the computational complexity of the LPA-FICI algorithm from 3 to N_t times. Comparison of the filter support selection between the original and the here-proposed algorithm is illustrated in Fig. 4.7. As shown in Fig. 4.7, the here-proposed algorithm selects the adaptive filter support size by detecting the signal edges and the corresponding signal regions, and for all signal samples inside the region, the same filter support size is used, unlike the original ICI rule which does not use previously calculated confidence intervals.

Hence, the main advantage of the here-proposed algorithm is its complexity reduction. For an N_t sample signal, the original ICI algorithm calculates up to N_t confidence intervals for each signal sample. Thus, the ICI algorithm can reach up to $N_t \times N_t = N_t^2$ confidence

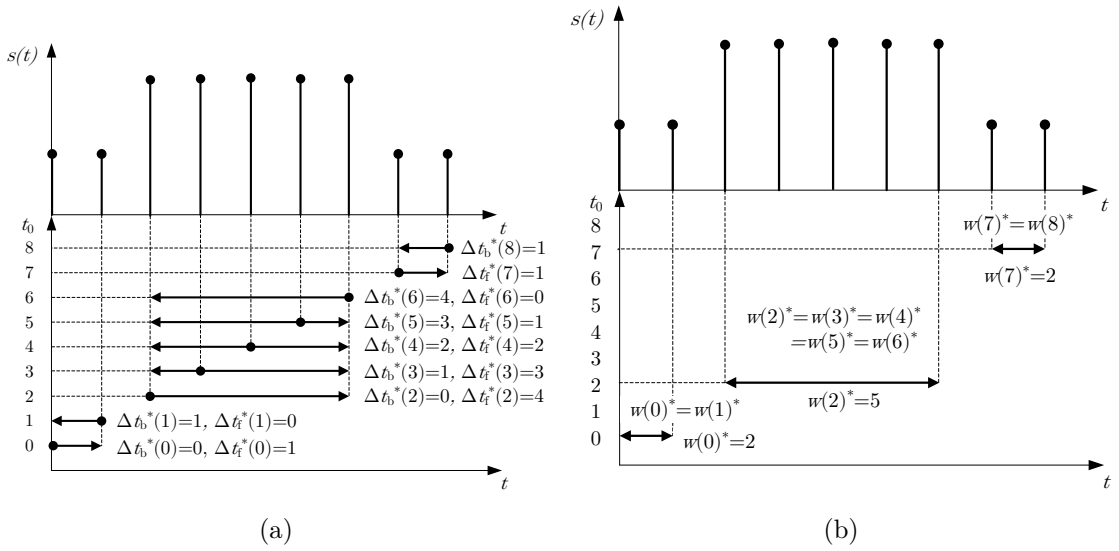


Figure 4.7: Example of the filter support selection using: (a) ICI algorithm (the algorithm calculates filter support size for each sample individually, resulting in up to N_t^2 calculated confidence intervals in total), (b) FICI algorithm (the algorithm calculates filter support in groups, resulting in exactly N_t calculated confidence intervals).

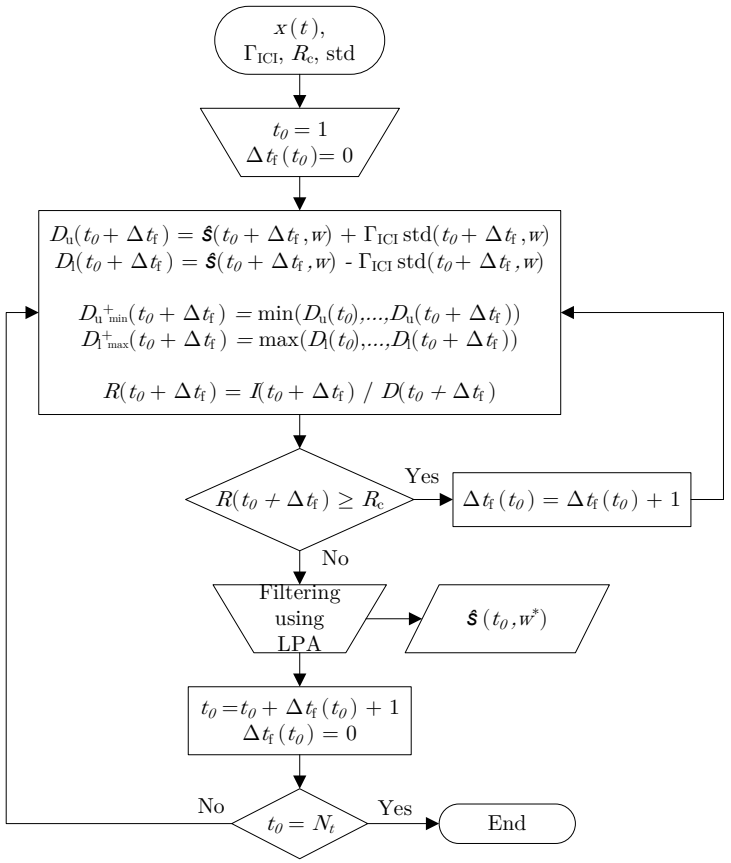


Figure 4.8: Flowchart of the here-proposed FICI algorithm.

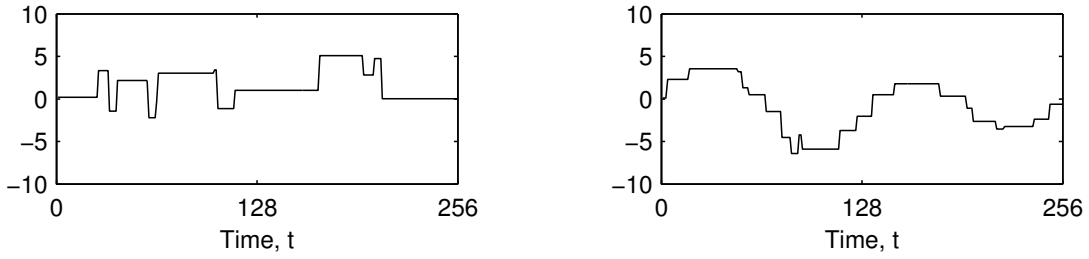


Figure 4.9: Denoised test signals with the LPA-FICI algorithm ($\Gamma_{\text{ICI}} = 3$, $R_c = 0.8$): (a) *Blocks* signal, (b) *HeaviSine* signal.

intervals calculation for the whole signal (as shown in Fig. 4.7(a)), which are then used for the adaptive filter support size selection. On the other hand, the here-proposed algorithm, which calculates the filter support size for the first sample of the corresponding signal region, requires only calculation of one confidence interval per sample, ergo N_t confidence intervals in total (as shown in Fig. 4.7(b)). Accordingly, the here-proposed algorithm reduces the number of computations from $\mathcal{O}(N_t^2)$ to $\mathcal{O}(N_t)$, hence the execution time is reduced by up to N_t times, when compared to the original ICI algorithm. Additionally, the here-proposed algorithm's number of operations does not depend on the signal shape, only its length as shown in (4.26), thus its execution time can be calculated, which is not the case for the original ICI algorithm. Furthermore, the ICI algorithm for any sample t_0 requires the knowledge of $\Delta t_{\text{f}}^*(t_0)$ next sample values, which hinders its use in real-time applications. On the other hand, the here-proposed algorithm does not have such requirement, hence being suitable in real-time signal processing applications.

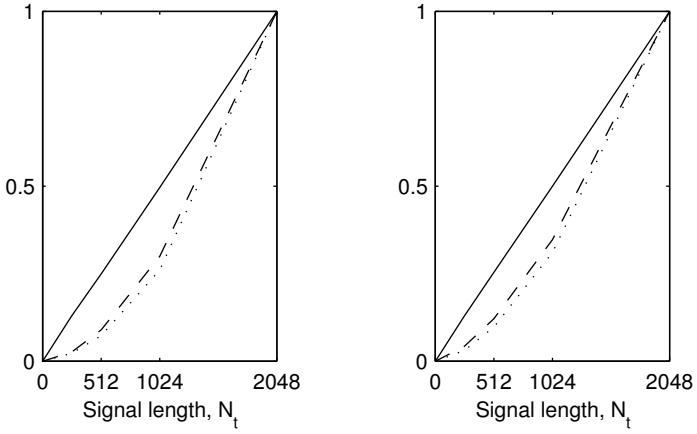
Fig. 4.8 gives the flowchart of the LPA-FICI algorithm. When compared to Fig. 4.3, where the flowchart of the original LPA-ICI is shown, it can be observed that the FICI algorithm eliminates the need for backward calculation of the filter support size, making the algorithm much simpler, and leading to the algorithm execution time reduction.

Example 4.4. In this example the LPA-FICI algorithm denoising performance has been tested, with identical simulation set-up as in Examples 4.2 and 4.3, making obtained results mutually comparable. The LPA-FICI algorithm parameters have been set as: $\Gamma_{\text{ICI}} = 3$ and $R_c = 0.8$. The obtained denoised signals are shown in Fig. 4.9, and denoising results are summarized in Table 4.3.

When compared to the LPA-ICI and the LPA-RICI execution times, shown in Tables 4.1 and 4.2, the LPA-FICI execution times are significantly lower: 40 times faster for signals with $N_t = 256$ samples, and 220 times faster for signals with $N_t = 2048$ samples. Furthermore, the LPA-FICI algorithm eliminates two main problems of the LPA-ICI algorithm: the execution times are constant across all signal types with the same length, and the algorithm execution time linearly increases with respect to the signal length (as

Table 4.3: Signal denoising results of the LPA-FICI algorithm with $\Gamma_{\text{ICI}} = 3$ and $R_c = 0.8$.

	$N_t = 256$		$N_t = 512$		$N_t = 1024$		$N_t = 2048$	
	Blocks	*HeaviSine*	*Blocks*	*HeaviSine*	*Blocks*	*HeaviSine*	*Blocks*	*HeaviSine*
MSE	0.2784	0.4166	0.2076	0.2668	0.1632	0.1742	0.1163	0.1156
RMSE	0.5222	0.6442	0.4526	0.5156	0.4021	0.4168	0.3396	0.3395
MAE	0.2329	0.5095	0.1733	0.4055	0.1371	0.3196	0.0981	0.2568
MAX	4.1416	2.3126	4.2513	2.3434	4.3791	2.5774	4.6255	2.7867
t [ms]	0.0015	0.0015	0.0029	0.0029	0.0058	0.0058	0.0118	0.0116



(a) (b)

Figure 4.10: Normalized algorithm execution times for the LPA-ICI (dotted line), the LPA-RICI (dashed line), and the LPA-FICI (solid line) algorithms: (a) *Blocks* signal, (b) *HeaviSine* signal. Due to the differences in order of magnitude between the algorithms execution times have been normalized for each algorithm separately.

shown in Fig. 4.10). On the other hand, denoising quality is not significantly disturbed: the LPA-FICI performs better then the LPA-ICI algorithm, and slightly worse than the LPA-RICI algorithm.

4.3 Estimation of Instantaneous Number of Time-Frequency Components Based on the Localized Rényi Entropy Information

Recently, a new method which estimates the instantaneous number of signal TF components based on the localized Rényi entropy has been proposed [71, 72]. The Rényi entropy, given by (2.37), is indifferent to the cross-terms when α_{R} is chosen as an odd integer value, i.e. integration of the cross-terms over the entire TF plane gives zero. This is why the Rényi entropy can be used to calculate the signal complexity, and by exploiting the counting property of the Rényi entropy, the number of instantaneous signal components, can be extracted by comparing the localized Rényi entropy of the TFD in question with a localized Rényi entropy of the TFD with a known number of signal components.

In other words, the instantaneous number of signal components, $n_c(t_0)$, is calculated as [71, 72]:

$$n_c(t_0) = 2^{R_z^{\alpha_R}(\rho_{z_{t_0}}(t,f)) - R_z^{\alpha_R}(\rho_{\text{ref}_{t_0}}(t,f))}, \tag{4.27}$$

where t_0 is the observed time slice, $\rho_{\text{ref}}(t,f)$ is a TFD of the reference signal, and the subscript t_0 in the TFD notation denotes that all samples of the TFD are set to zero, except the samples in vicinity of t_0, that is:

$$\rho_{z_{t_0}}(t,f) = \begin{cases} \rho_z(t,f), & t_0 - \Delta t < t < t_0 + \Delta t, \\ 0, & \text{otherwise}, \end{cases} \tag{4.28}$$

where Δt is the user defined localization parameter defining a length of the observed time interval.

The reference signal can be chosen arbitrarily; in [71, 72] the reference signal is chosen to be a stationary cosine signal with a normalized frequency 0.1, and an amplitude of 1. In the conducted simulations, it seems that the selection of the reference signal does not play a significant role in the overall performance of the algorithm. On the other hand, the selection of the appropriate TFD plays a crucial role. In order to guaranty a meaningful solution of the algorithm, $\rho_z(t,f)$ and $\rho_{\text{ref}}(t,f)$ have to be calculated in the same way. In [71] it has been shown that the extended modified B distribution (its kernel function is given in Table 2.2) is a good choice for the TFD, as substantiated by the TFD comparisons performed in Examples 2.4 - 2.7.

The flowchart of the algorithm is shown in Fig. 4.11. The two additional steps ensure the robustness of the algorithm: firstly, the TFDs are thresholded by removing 10% of the samples with the smallest amplitude; and secondly, if energy of the current time slice, $E_{z_{t_0}}$, is smaller then the predefined minimum energy threshold, $E_{\min}$, that is [71]:

$$\underbrace{\sum^{N_f} \rho_z(t_0,f)}_{E_{z_{t_0}}} < \underbrace{\frac{0.001 N_t}{\Delta t} \sum^{N_t} \sum^{N_f} \rho_z(t,f)}_{E_{\min}}, \tag{4.29}$$

the number of signal components in the current time slice, $n_c(t_0)$, is set to zero.

Example 4.5. In this example, the instantaneous number of components for the previously defined signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$ has been calculated. The used reference signal is a stationary cosine signal with a normalized frequency 0.1, and an amplitude of 1, as in [71, 72], while the selected TFD is the extended modified B distribution with the kernel parameters set as $\alpha_{\text{EMB}} = 0.1$ and $\beta_{\text{EMB}} = 0.12$. The order of the Rényi entropy is set to $\alpha_{\text{R}} = 7$, with the localization parameter $\Delta t = 11$.

The resulting instantaneous number of components, along with the extended modified

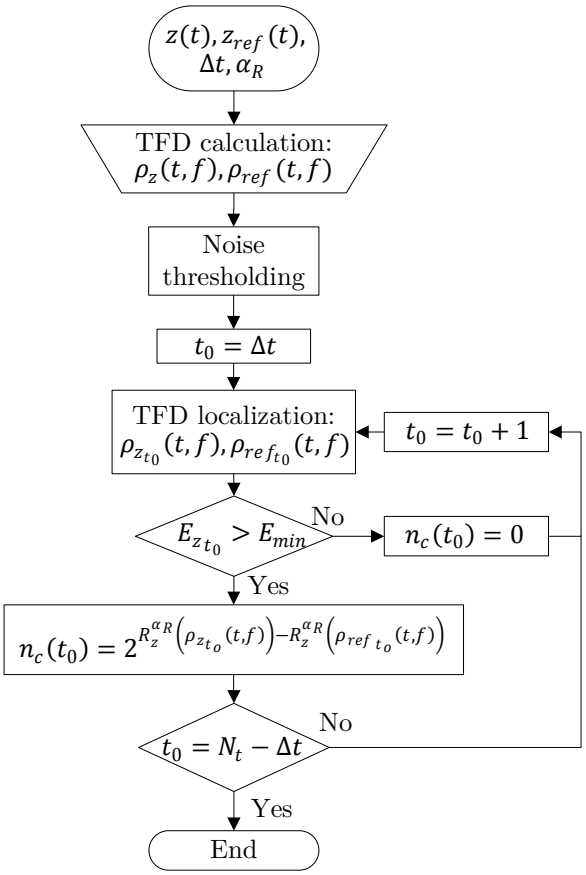


Figure 4.11: Flowchart of the localized Rényi entropy based algorithm for estimation of the instantaneous number of signal components.

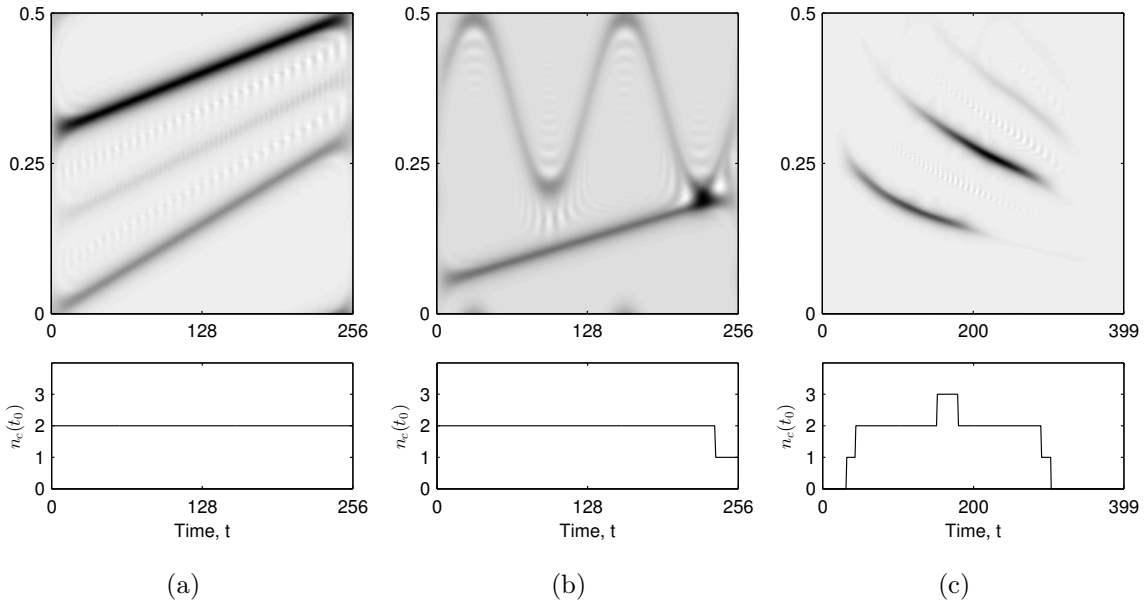


Figure 4.12: The instantaneous number of signals components calculated from the localized Rényi entropy of the extended modified B distribution ($\alpha_{\text{EMB}} = 0.1, \beta_{\text{EMB}} = 0.12$) of the: (a) three LFM component signal, $z_{\text{LFM}}(t)$, (b) two component non-LFM signal, $z_{\text{nonLFM}}(t)$, (c) bat echolocation signal, $z_{\text{bat}}(t)$.

B distribution of the considered signals are shown in Fig. 4.12. For the signal $z_{\mathrm{LFM}}(t)$, it can be seen that the algorithm failed to detect the weakest signal component, which is masked with the cross-terms between the remaining two components. On the other hand, for the signal $z_{\mathrm{nonLFM}}(t)$, it can be seen that the algorithm has correctly calculated the instantaneous number of signal components, with an exception of time slices where the two components are very closely located. This problem is solved in [72], where the signal component intersections are detected from the derivative of $n_c(t_0)$. For the signal $z_{\mathrm{bat}}(t)$, it can be seen that the algorithm has correctly detected the instantaneous number of signal components.

4.4 Summary

In this chapter, the three adaptive methods have been discussed, which when applied in the CS based TFD calculation will enhance the concentration of the resulting sparse TFD.

The first method, which detects the intersection between the cross-terms and the respective AF axis, removes the need for experimental CS-AF area selection, providing the signal adaptive method which avoids the cross-term inclusion in the reconstructed sparse TFD. The effectiveness of the here-proposed method has been tested by constructing the standard TFD kernel and comparing the concentration of the resulting TFDs with the concentration of the RGK-TFDs. The conducted simulations on synthetical and real-life signals have shown that the proposed kernel performs competitively with the RGK when the considered signal is composed from the LFM components; however, when the considered signal has a highly nonlinear frequency modulated component, the proposed kernel fails to properly capture the auto-terms in its pass-band.

The second method, namely the FICI rule, provides a method for the adaptive sparse TFD thresholding by searching the sparse TFD for samples with the similar values. The denoising capabilities of the proposed LPA-FICI method have been tested on the synthetical signals, and its performance has been compared with the performance of the LPA-ICI method and the LPA-RICI method. The conducted experiments have shown that the LPA-FICI convergence rate is significantly faster, while preserving the same denoising quality.

The third method, based on the localized Rényi entropy, detects the instantaneous number of signal components in the TFD. This information can then be used to estimate the TFD sparsity level. The method has been used in order to calculate the instantaneous number of components present in synthetical and real-life signals. The conducted simulation have shown the satisfactory performance of the method.

Chapter 5

Compressive Sensing Based Time-Frequency Distribution Concentration Enhancement

In this chapter, the three methods for concentration enhancement of the sparse TFD are discussed. The first method, based on the APK, previously discussed in Section 4.1, adaptively detects the largest possible rectangular or elliptical CS-AF area without the cross-term inclusion. The goal of the here-proposed CS-AF selection method is to provide more cross-term free samples to the sparse signal reconstruction algorithm, resulting in a more accurate reconstruction results and a faster algorithm convergence rate. The remaining two here-proposed methods are sparse reconstruction algorithms: the first sparse reconstruction algorithm is based on the FICI rule, previously discussed in Section 4.2.4, while the second sparse reconstruction algorithm is based on the localized Rényi entropy, previously discussed in Section 4.3. The here-proposed adaptive methods are tested on the synthetical and real-life signals, while the obtained results are compared with the results obtained with the currently available state-of-the-art algorithms for the sparse signal reconstruction.

5.1 Adaptive Compressive Sensing Area Selection of the Ambiguity Function

As previously discussed in Chapter 2, the cross-terms are located away from the AF domain origin with the distance equal to the time-frequency distance between the components, leading to a conclusion that selecting the CS-AF area independent of AF geometry is suboptimal; when dealing with a closely located components, the CS-AF area is limited by the cross-terms location; however, when components are more spread-out, the

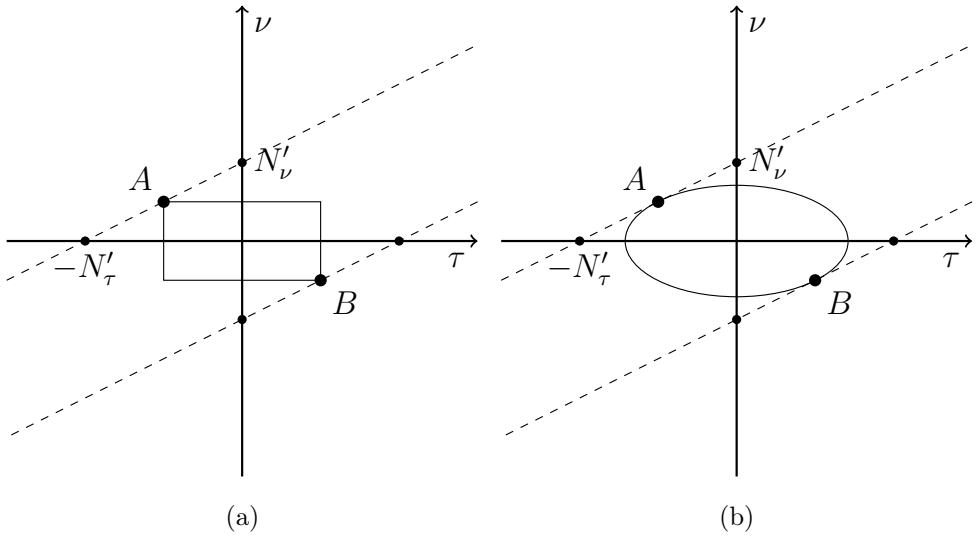


Figure 5.1: The adaptive CS-AF sensing area geometry: (a) the rectangular mask, (b) the elliptical mask. Dashed lines represent the cross-terms pair which is closest to the AF domain origin, while the points $A(-N_\tau/2, N_\nu/2)$ and $B(N_\tau/2, -N_\nu/2)$ are touching points between the CS-AF sensing mask and the cross-terms line.

CS-AF area can be significantly larger. This has served as a motivation for development of a here-proposed method for the adaptive CS-AF area selection, with the general goal to capture as large as possible area around the AF origin, without including any of the cross-terms, which if included would reappear in the sparse TFD. The reason behind this specific approach is to lower the requirements of the reconstruction algorithm; it is easier to reconstruct a TFD having more AF auto-term samples to begin with.

The here-proposed method constructs the CS-AF sensing mask as a $N'_\nu \times N'_\tau$ rectangle centered around the AF domain origin, that is:

$$\boldsymbol{\phi}(\nu,\tau) = \begin{cases} 1, & \nu \leq \left|\dfrac{N'_\nu}{2}\right|, \tau \leq \left|\dfrac{N'_\tau}{2}\right|, \\ 0, & \text{otherwise}, \end{cases} \tag{5.1}$$

where the parameters N'_ν and N'_τ are obtained using the algorithm proposed in Section 4.1. The algorithm is implemented in a MATLAB function `fun_adaptiveCSAF.m`, the code of which is given in Appendix B. The rectangular CS-AF sensing mask constructed in this way will have two of its vertices on the cross-terms line, as shown in Fig. 5.1(a), ensuring the largest possible rectangular CS-AF area without the cross-term inclusion.

Alternatively, the adaptive CS-AF sensing mask can be constructed as an ellipse, centered around the AF domain origin, where the cross-terms line is a tangent to the elliptical CS-AF sensing mask at the point $A(-N'_\tau/2, N_\nu/2)$, as shown in Fig. 5.1(b). The two radiuses, r_τ and r_ν, are obtained from the intersection of the cross-terms line with the elliptical CS-AF sensing mask:

$$\nu = \frac{N'_\nu}{N'_\tau}\tau + N'_\nu, \tag{5.2a}$$

$$\frac{\tau^2}{r_\tau^2} + \frac{\nu^2}{r_\nu^2} = 1, \tag{5.2b}$$

leading to the following quadratic equation:

$$\tau^2\left(r_\nu^2 + r_\tau^2\frac{N'^2_\nu}{N'^2_\tau}\right) + \tau\left(2r_\tau^2\frac{N'^2_\nu}{N'_\tau}\right) + (r_\tau^2 N'^2_\nu - r_\tau^2 r_\nu^2) = 0. \tag{5.3}$$

The cross-terms line will be a tangent to the elliptical CS-AF sensing mask when there is only a single intersection point between them, thus the determinant of (5.3) has to be zero, leading to the following expression:

$$\frac{r_\tau^2}{N'^2_\tau} + \frac{r_\nu^2}{N'^2_\nu} = 1, \tag{5.4}$$

which further leads to a unique solution of the quadratic equation (5.3):

$$\tau = \frac{-r_\tau^2}{N'_\tau\left(\frac{r_\tau^2}{N'^2_\tau} + \frac{r_\nu^2}{N'^2_\nu}\right)} = \frac{-r_\tau^2}{N'_\tau}. \tag{5.5}$$

By plugging the desired intersection point between the cross-terms line and the elliptical CS-AF sensing mask as $\tau = -N'_\tau/2$ and $\nu = N'_\nu/2$ into (5.5) and (5.2b), the two radiuses of the elliptical CS-AF sensing mask, r_τ and r_ν, are obtained as:

$$r_\tau = \frac{N'_\tau}{\sqrt{2}}, \qquad r_\nu = \frac{N'_\nu}{\sqrt{2}}, \tag{5.6}$$

leading to a final expression for the elliptical CS-AF sensing mask:

$$\boldsymbol{\phi}(\nu,\tau) = \begin{cases} 1, & \dfrac{2\nu^2}{N'^2_\nu} + \dfrac{2\tau^2}{N'^2_\tau} \leq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

By comparing the rectangular CS-AF area containing $N'_\tau N'_\nu$ samples, and the elliptical CS-AF area containing $r_\tau r_\nu \pi = N'_\tau N'_\nu \pi/2$ samples, it can be concluded that the elliptical CS-AF sensing mask is $\pi/2$ times larger than the rectangular CS-AF sensing mask, allowing even more samples to be included from which the sparse TFD is reconstructed.

Example 5.1. In this example, the here-proposed adaptive CS-AF areas have been calculated for the previously defined signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$. Following the standard CS-AF methodology, the following CS-AF areas are obtained: $N'_\tau = N'_\nu = 15$, for the signals $z_{\text{LFM}}(t)$ and $z_{\text{nonLFM}}(t)$, while $N'_\tau = N'_\nu = 19$ for the signal $z_{\text{bat}}(t)$. On the other hand, the here-proposed adaptive CS-AF area selection method has resulted in

$N'_\tau = 17, N'_\nu = 73$ for the signal $z_{\mathrm{LFM}}(t)$, $N'_\tau = 15, N'_\nu = 39$ for the signal $z_{\mathrm{nonLFM}}(t)$, and $N'_\tau = 21, N'_\nu = 71$ for the signal $z_{\mathrm{bat}}(t)$. The resulting rectangular CS-AF sensing masks are shown in Figs. 5.2(a) - 5.2(c), while the elliptical CS-AF sensing masks are shown in Figs. 5.2(d) - 5.2(f). Note that the IHT_λ, NIHT_λ, and FPC algorithm for the signal $z_{\mathrm{bat}}(t)$ did not converge, producing NaN solution, and this is why the respective sparse reconstruction results have been omitted from the respective figures and tables.

When compared to the manually selected CS-AF sensing masks, the adaptively detected rectangular CS-AF sensing mask is 5 times larger for the signal $z_{\mathrm{LFM}}(t)$, 2.5 times larger for the signal $z_{\mathrm{nonLFM}}(t)$, and 4 times larger for the signal $z_{\mathrm{bat}}(t)$. Further enlargement of the CS-AF area, by including approximately 50% more AF samples, is obtained with the adaptively detected elliptical CS-AF sensing mask, achieving $N_{\mathrm{CS}} = 1935$ samples for the signal $z_{\mathrm{LFM}}(t)$, $N_{\mathrm{CS}} = 923$ samples for the signal $z_{\mathrm{nonLFM}}(t)$, and $N_{\mathrm{CS}} = 2380$ samples for the signal $z_{\mathrm{bat}}(t)$. Furthermore, by visual inspection of Fig. 5.2 it can be

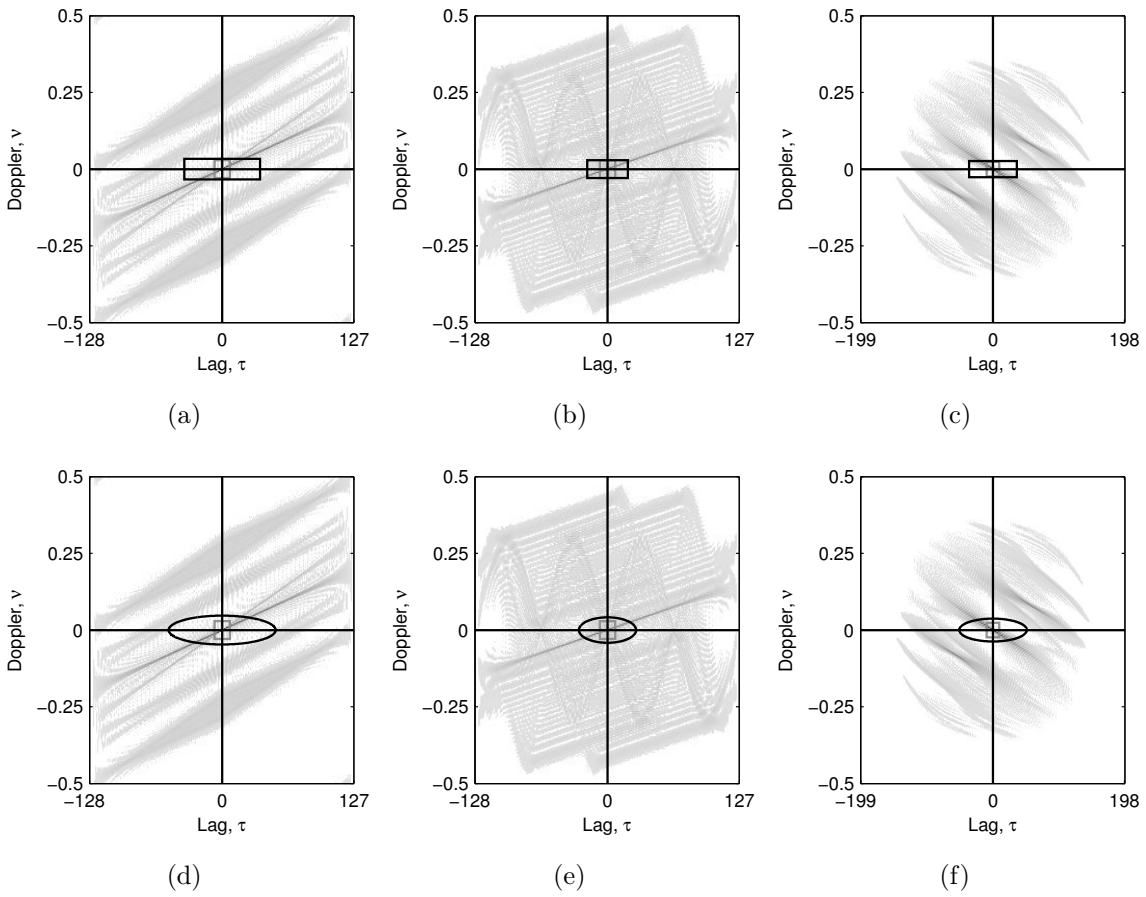


Figure 5.2: The adaptively detected CS-AF sensing mask: (a) the rectangular CS-AF sensing mask for the signal $z_{\mathrm{LFM}}(t)$ ($N'_\tau = 17, N'_\nu = 73$), (b) the rectangular CS-AF sensing mask for the signal $z_{\mathrm{nonLFM}}(t)$ ($N'_\tau = 15, N'_\nu = 39$), (c) the rectangular CS-AF sensing mask for the signal $z_{\mathrm{bat}}(t)$ ($N'_\tau = 21, N'_\nu = 71$), (d) the elliptical CS-AF sensing mask for the signal $z_{\mathrm{LFM}}(t)$ ($N_{\mathrm{CS}} = 1935$), (e) the elliptical CS-AF sensing mask for the signal $z_{\mathrm{nonLFM}}(t)$ ($N_{\mathrm{CS}} = 923$), (f) the elliptical CS-AF sensing mask for the signal $z_{\mathrm{bat}}(t)$ ($N_{\mathrm{CS}} = 2380$). The gray square is a manually selected CS-AF sensing mask with $N'_\tau = N'_\nu = 15$ for the signals $z_{\mathrm{LFM}}(t)$ and $z_{\mathrm{nonLFM}}(t)$, and $N'_\tau = N'_\nu = 19$ for the signal $z_{\mathrm{bat}}(t)$.

concluded that the performance of the here-proposed CS-AF area selection algorithm has fulfilled the previously discussed design goals of the adaptive rectangular and elliptical CS-AF sensing mask.

Next, the performance of the ℓ_0 norm and the ℓ_1 norm based sparse reconstruction algorithms with the adaptively detected CS-AF sensing mask has been compared with the performance of the sparse reconstruction algorithms when the CS-AF sensing mask is selected manually. The simulation set-up is the same as in Example 3.2 for the ℓ_0 norm based reconstruction algorithms, and same as in Example 3.3 for the ℓ_1 norm based reconstruction algorithms, making the previously obtained results presented in Tables 3.1 and 3.2 comparable with the results obtained in this example. The resulting sparse TFDs based on the ℓ_0 norm minimization are shown in Figs. 5.3 - 5.5, with the results summarized in Table 5.1 for the adaptively detected rectangular CS-AF sensing mask and in Table 5.2 for the adaptively detected elliptical CS-AF sensing mask. The resulting sparse TFDs based on the ℓ_1 norm minimization are shown in Figs. 5.6 - 5.8, with the results summarized in Table 5.3 for the adaptively detected rectangular CS-AF sensing mask and in Table 5.4 for the adaptively detected elliptical CS-AF sensing mask.

By visual inspection of the obtained sparse TFDs based on the ℓ_0 norm minimization and the adaptively detected CS-AF sensing mask (shown in Figs. 5.3 - 5.5), it can be concluded that the newly obtained ℓ_0 norm based TFDs are significantly better concentrated than the previously obtained ℓ_0 norm based sparse TFDs with the manually selected CS-AF sensing mask (shown in Figs. 3.3 - 3.5), and are in fact, comparable with the sparse TFDs obtained by the ℓ_1 norm minimization and the manually selected CS-AF sensing mask (shown in Figs. 3.6 - 3.8). The problem with the weakest signal component being hard-thresholded, previously discussed in Example 3.2, has been partially solved (since more AF samples are processed by the sparse reconstruction algorithm), but it is still present. This can be seen by comparing the ℓ_0 norm based sparse TFDs of the signals $z_{\mathrm{LFM}}(t)$ and $z_{\mathrm{bat}}(t)$ obtained with the manually selected CS-AF sensing mask (shown in Figs. 3.3 and 3.5) with the respective TFDs obtained with the adaptively detected CS-AF sensing mask (shown in Figs. 5.3 and 5.5), as the weakest component is present in the more newly obtained ℓ_0 norm based sparse TFDs with the adaptively detected CS-AF sensing mask. The similar problem is present in the ℓ_0 norm based sparse TFDs of the signal $z_{\mathrm{nonLFM}}(t)$ where the sinusoidal frequency modulated component gets hard-thresholded because there is not enough CS-AF samples corresponding to the signal component in question.

Furthermore, the concentration measure and the Rényi entropy of the newly obtained ℓ_0 norm based sparse TFDs, shown in Table 5.1, achieved approximately $5\% - 10\%$ improvement when compared to the respective sparse TFDs with the manually selected CS-AF sensing mask. The problem with the concentration measure being unproportionately low when a signal component is missing, discussed in Example 3.2, is also present

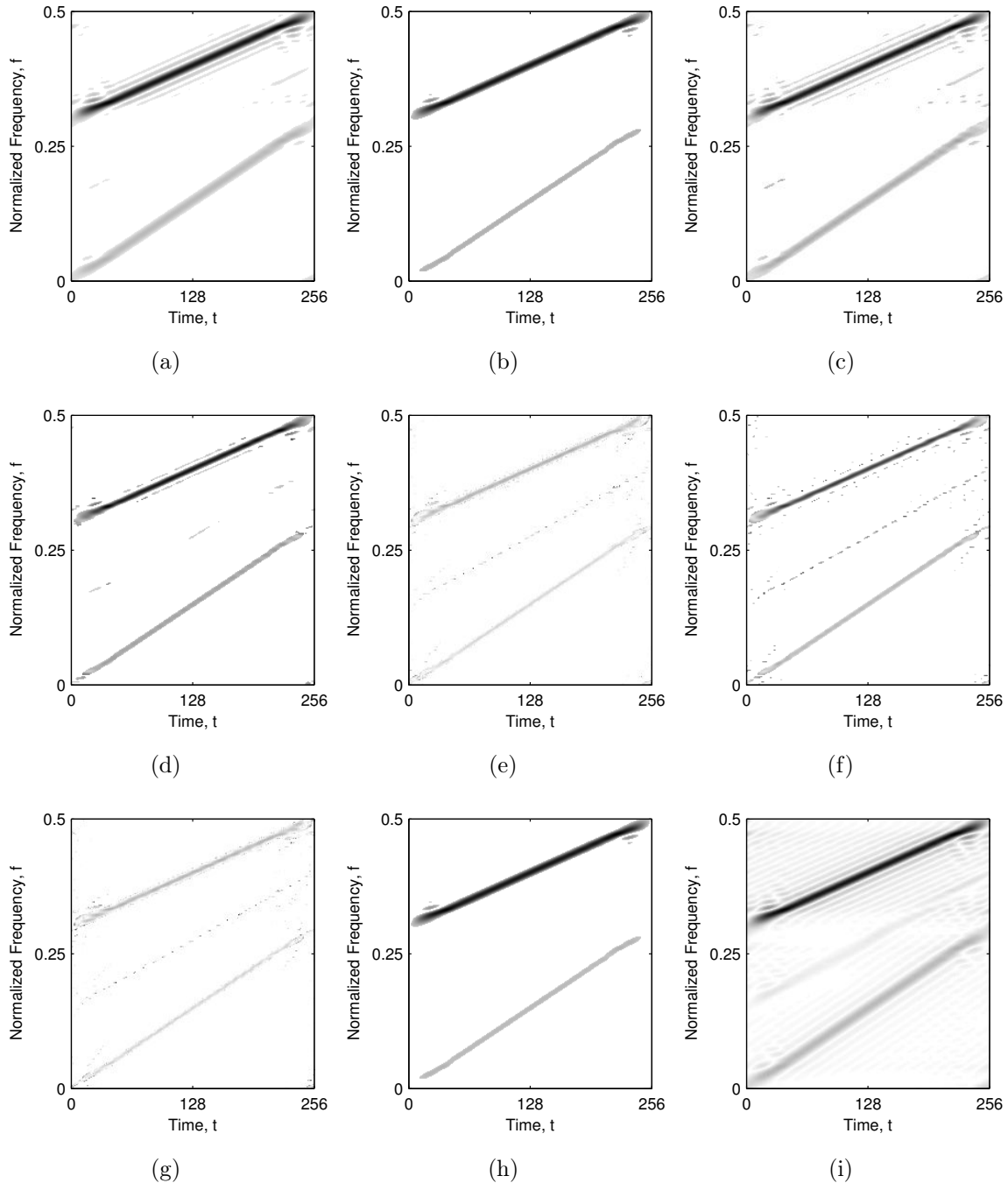


Figure 5.3: The reconstructed sparse TFDs based on the ℓ_0 norm minimization of the three LFM component signal $z_{\mathrm{LFM}}(t)$ with the adaptively detected rectangular CS-AF sensing mask: (a) IHT_{λ}, (b) IHT_{K}, (c) NIHT_{λ}, (d) NIHT_{K}, (e) AIHT_{K}, (f) ECME_{K}, (g) DORE_{K}, (h) HTP_{K}, (i) FHTP_{K} algorithm.

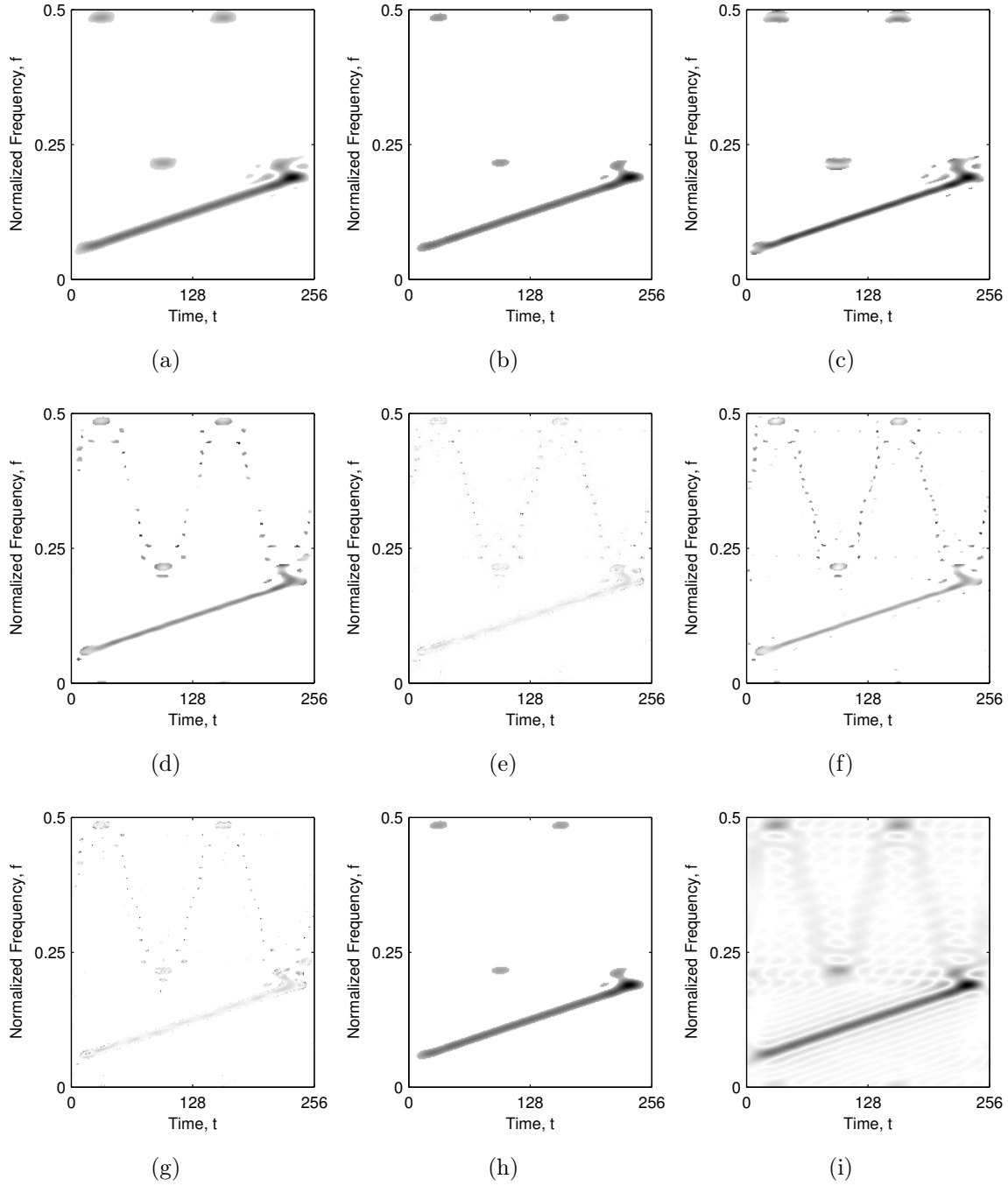


Figure 5.4: The reconstructed sparse TFDs based on the ℓ_0 norm minimization of the two component signal $z_{\mathrm{nonLFM}}(t)$ with the adaptively detected rectangular CS-AF sensing mask: (a) IHT_{λ}, (b) IHT_{K}, (c) NIHT_{λ}, (d) NIHT_{K}, (e) AIHT_{K}, (f) ECME_{K}, (g) DORE_{K}, (h) HTP_{K}, (i) FHTP_{K} algorithm.

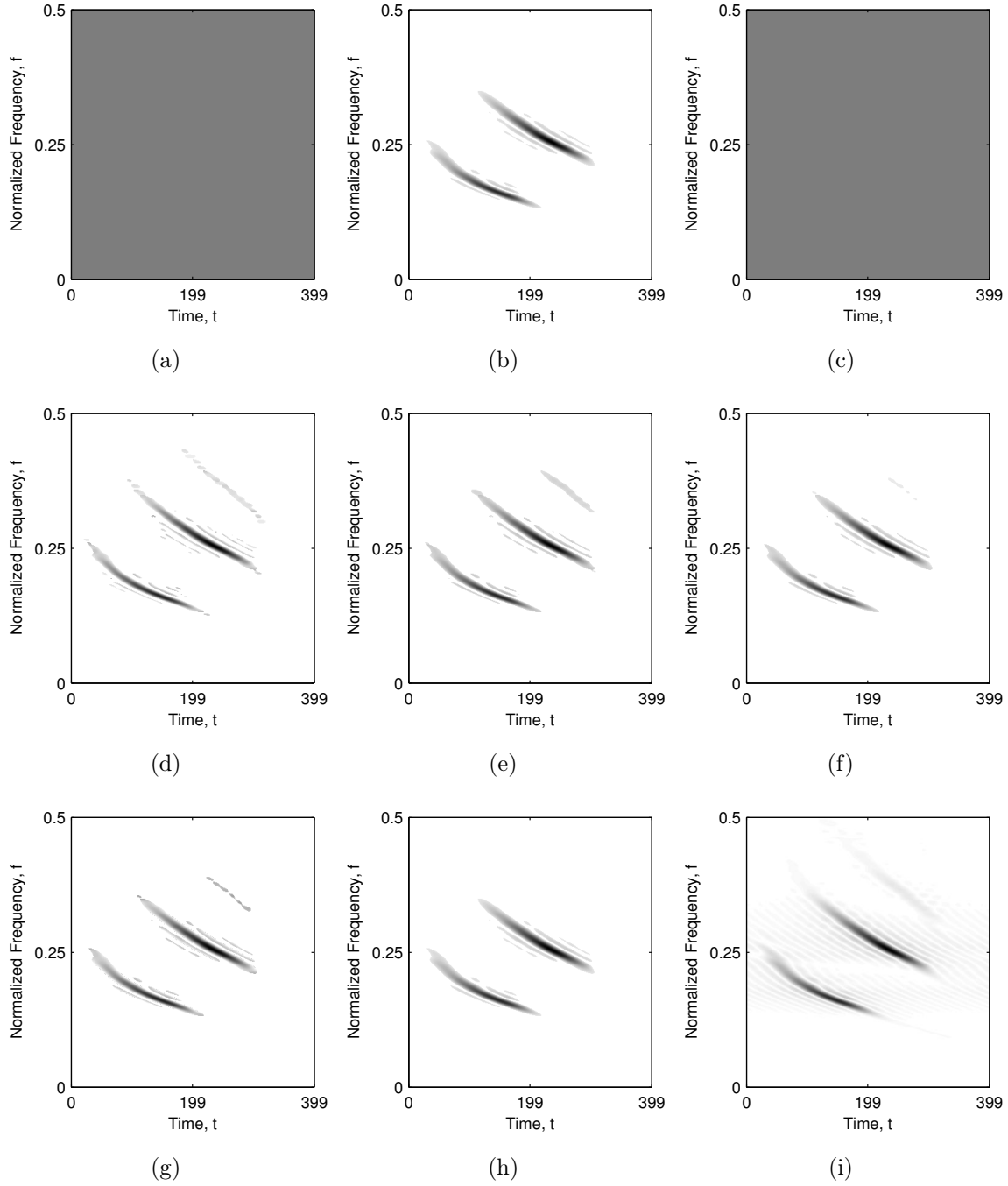


Figure 5.5: The reconstructed sparse TFDs based on the ℓ_0 norm minimization of the bat echolocation signal $z_{\mathrm{bat}}(t)$ with the adaptively detected rectangular CS-AF sensing mask: (a) IHT_{λ}, (b) IHT_{K}, (c) NIHT_{λ}, (d) NIHT_{K}, (e) AIHT_{K}, (f) ECME_{K}, (g) DORE_{K}, (h) HTP_{K}, (i) FHTP_{K} algorithm.

Table 5.1: Rényi entropies, $R_z^{\alpha_R}$, concentration measures, M_z^S, and the algorithm execution times of the reconstruction algorithms based on the ℓ_0 norm minimization with the adaptively detected rectangular CS-AF sensing mask. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IHT_λ	12.2219	0.1250	**3.6448**	11.5645	0.0576	**3.6191**	–	–	–
IHT_K	11.4876	0.0556	61.3573	11.1465	0.0384	55.4814	12.2063	0.0449	47.3696
NIHT_λ	12.1530	0.1122	14.2751	11.2996	0.0468	35.5876	–	–	–
NIHT_K	11.4655	0.0556	85.1035	10.9638	0.0372	212.7585	**12.1501**	**0.0444**	277.1088
AIHT_K	11.1962	0.0525	1675.9107	9.9527	0.0336	4068.4165	12.2702	0.0458	33.7429
ECME_K	11.3839	0.0544	534.1264	10.8110	0.0359	695.4658	12.2160	0.0450	64.1194
DORE_K	**11.1127**	**0.0519**	1978.9220	**9.8552**	**0.0327**	2979.5565	12.2225	0.0447	96.4435
HTP_K	11.4564	0.0551	21.2135	11.1283	0.0384	20.5624	12.2052	0.0449	78.6640
FHTP_K	13.1762	0.6142	8.0521	13.4477	0.6808	8.0569	13.5272	0.4687	31.5296

Table 5.2: Rényi entropies, $R_z^{\alpha_R}$, concentration measures, M_z^S, and the algorithm execution times of the reconstruction algorithms based on the ℓ_0 norm minimization with the adaptively detected elliptical CS-AF sensing mask. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IHT_λ	11.9752	0.1106	**4.9174**	11.6856	0.0654	48.8216	–	–	–
IHT_K	11.3771	0.0548	27.9543	11.1189	0.0383	41.4901	12.0855	0.0440	56.3378
NIHT_λ	11.8141	0.0913	34.3817	11.5536	0.0582	36.1657	–	–	–
NIHT_K	11.3790	0.0551	82.2913	11.0868	0.0380	129.2657	12.0869	0.0443	189.3486
AIHT_K	11.1786	0.0521	2742.5673	10.2029	0.0345	2444.3186	12.1569	0.0450	42.3939
ECME_K	11.2438	0.0526	861.2112	10.8339	0.0359	832.8246	12.1020	0.0442	72.7568
DORE_K	**11.1469**	**0.0518**	1918.2433	**9.9356**	**0.0326**	3578.9344	12.0937	**0.0436**	122.9286
HTP_K	11.3725	0.0547	23.5431	11.1093	0.0382	22.7235	**12.0836**	0.0439	84.8922
FHTP_K	12.7955	0.5513	9.6387	13.3113	0.6202	**9.5300**	13.3001	0.4859	**37.4837**

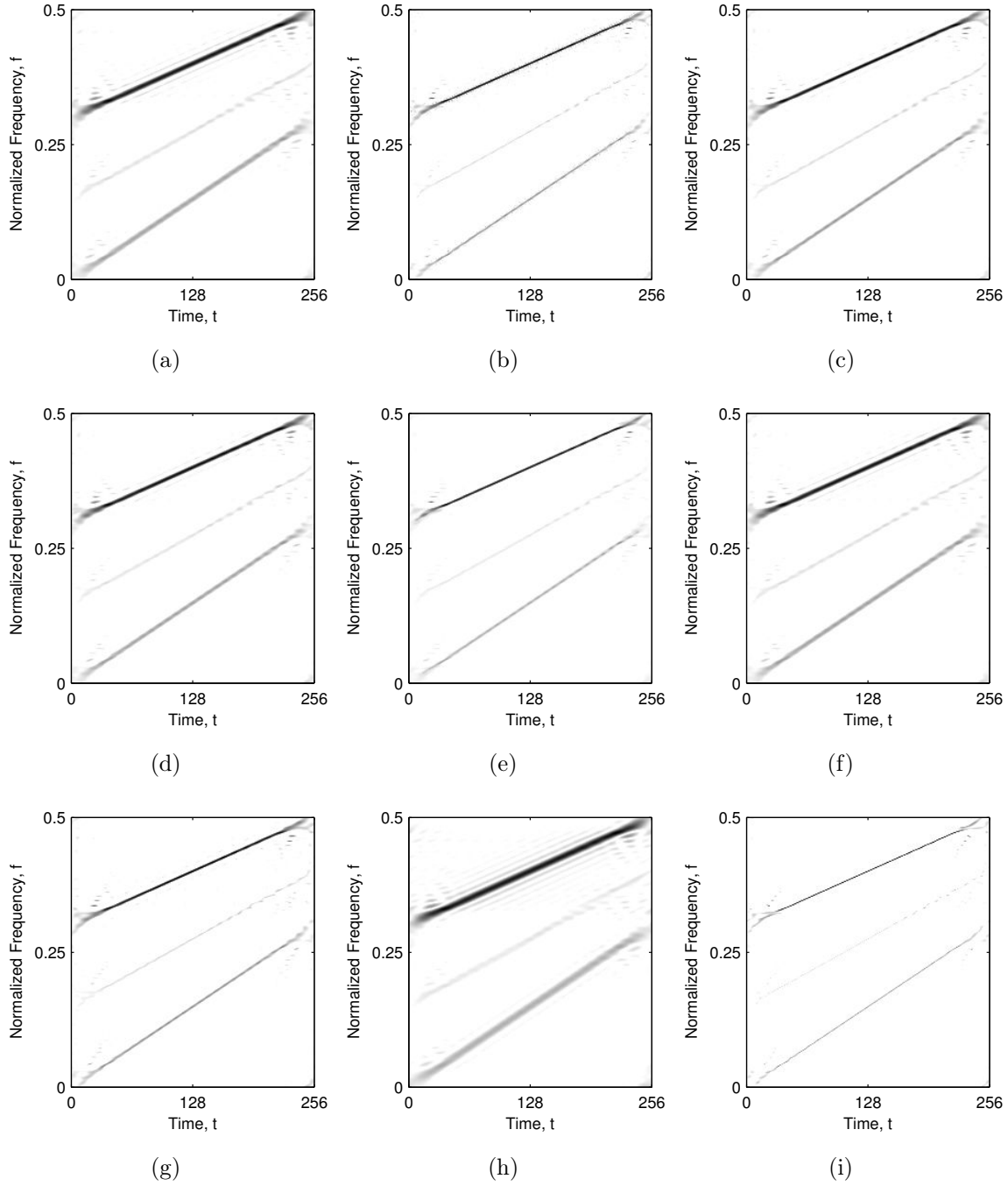


Figure 5.6: The reconstructed sparse TFDs based on the ℓ_1 norm minimization of the three LFM component signal $z_{\mathrm{LFM}}(t)$ with the adaptively detected rectangular CS-AF sensing mask: (a) IST, (b) TwIST, (c) FISTA, (d) SpaRSA, (e) FPC, (f) GPSR, (g) NESTA, (h) SALSA, (i) YALL1 algorithm.

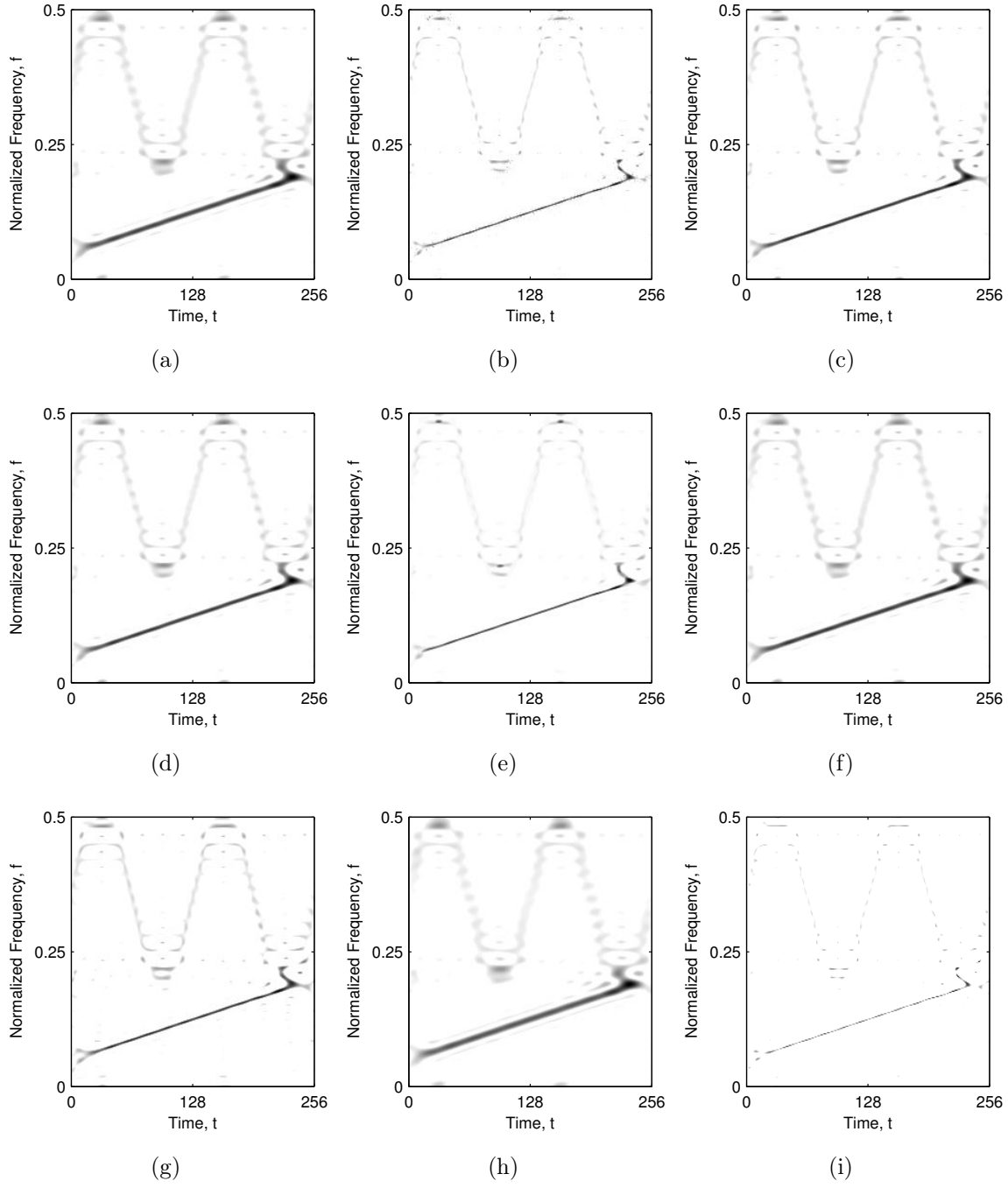


Figure 5.7: The reconstructed sparse TFDs based on the ℓ_1 norm minimization of the two component signal $z_{\mathrm{nonLFM}}(t)$ with the adaptively detected rectangular CS-AF sensing mask: (a) IST, (b) TwIST, (c) FISTA, (d) SpaRSA, (e) FPC, (f) GPSR, (g) NESTA, (h) SALSA, (i) YALL1 algorithm.

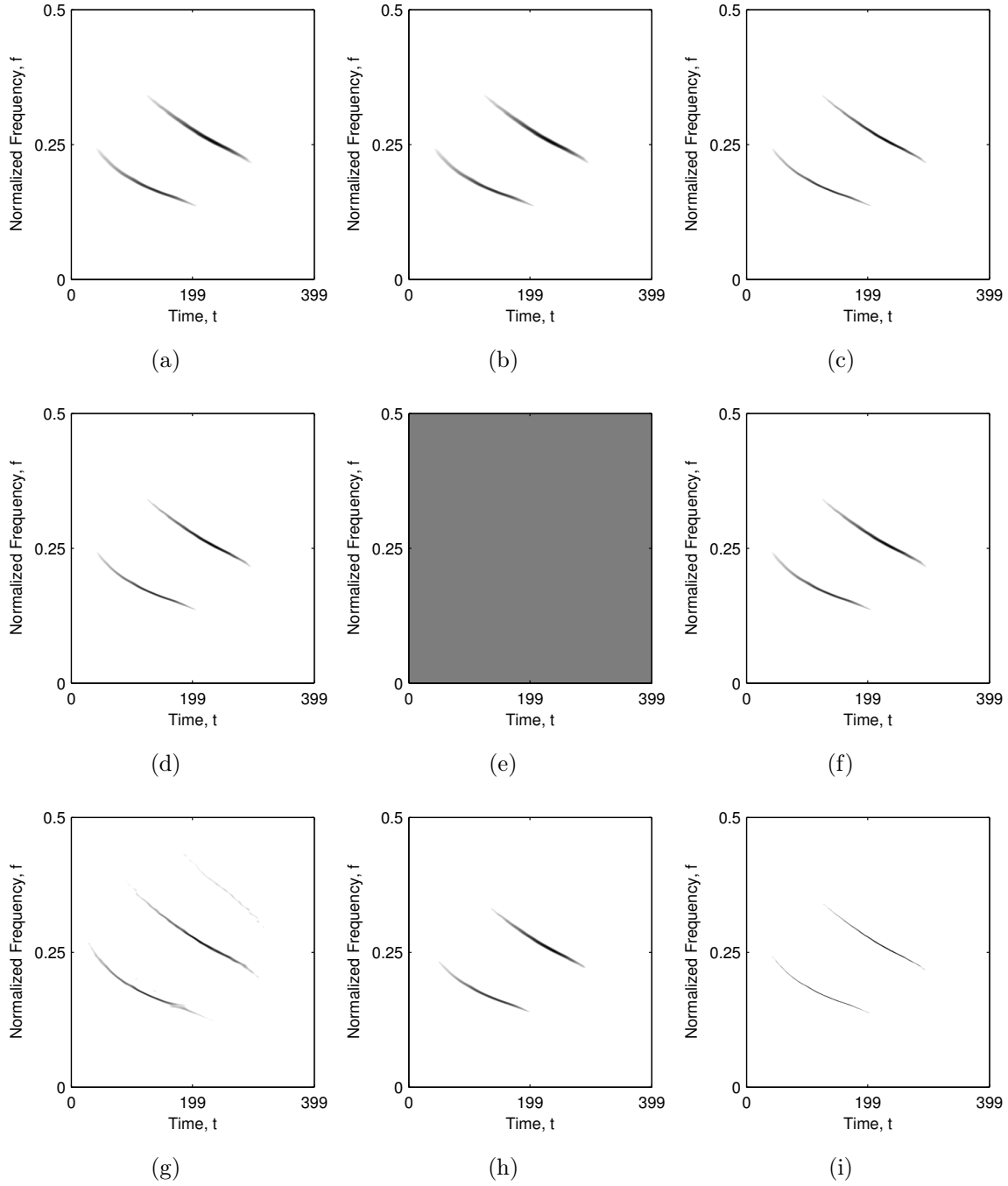


Figure 5.8: The reconstructed sparse TFDs based on the ℓ_1 norm minimization of the bat echolocation signal $z_{\mathrm{bat}}(t)$ with the adaptively detected rectangular CS-AF sensing mask: (a) IST, (b) TwIST, (c) FISTA, (d) SpaRSA, (e) FPC, (f) GPSR, (g) NESTA, (h) SALSA, (i) YALL1 algorithm.

Table 5.3: Rényi entropies, $R_z^{\alpha_R}$, concentration measures, M_z^S, and the algorithm execution times of the reconstruction algorithms based on the ℓ_1 norm minimization with the adaptively detected rectangular CS-AF sensing mask. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IST	11.5542	0.0992	406.7996	11.7681	0.1229	277.2688	10.5699	0.0131	606.3275
TwIST	10.5232	0.0560	184.6928	10.3268	0.0447	216.7444	10.6202	0.0139	457.2896
FISTA	10.9062	0.0586	276.6017	11.1545	0.0718	229.3851	10.1051	0.0093	579.2088
SpaRSA	11.1068	0.0750	126.5906	11.3850	0.0950	**79.1227**	10.1322	0.0095	348.1371
FPC	10.4645	0.0636	655.8862	10.6083	0.1008	824.9775	–	–	–
GPSR	11.3511	0.0828	292.2279	11.5828	0.1046	193.1485	10.2641	0.0106	**273.1314**
NESTA	10.6336	0.1227	1466.2038	11.0566	0.1870	1190.9418	10.9388	0.2530	2655.0842
SALSA	12.0660	0.1866	**116.6848**	11.9081	0.1629	153.0509	10.1379	0.0150	1219.6876
YALL1	**9.2721**	**0.0180**	2084.0243	**9.0030**	**0.0132**	2341.8449	**8.8022**	**0.0037**	10772.1951
ℓ_1-ls	9.4666	0.2366	6304.5451	9.6500	0.4230	6184.1323	13.4823	0.4664	4332.4106

Table 5.4: Rényi entropies, $R_z^{\alpha_R}$, concentration measures, M_z^S, and the algorithm execution times of the reconstruction algorithms based on the ℓ_1 norm minimization with the adaptively detected elliptical CS-AF sensing mask. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$	$R_z^{\alpha_R}$ $\alpha_R = 3$	M_z^S $p_s = 2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IST	11.4124	0.0987	542.1344	11.8659	0.1242	368.5609	10.4259	0.0118	854.2265
TwIST	**9.9994**	**0.0447**	313.9779	10.3686	**0.0488**	284.1496	**9.6863**	**0.0070**	462.7869
FISTA	10.6769	0.0541	416.9572	11.2028	0.0740	318.7749	9.9717	0.0084	780.7640
SpaRSA	10.8791	0.0708	227.2669	11.6143	0.1065	**82.6394**	10.0369	0.0089	433.2059
FPC	10.3455	0.0645	1020.6865	10.6763	0.0883	1118.3698	–	–	–
GPSR	11.2197	0.0831	370.3489	11.6748	0.1064	238.8542	10.1174	0.0095	**385.8460**
NESTA	10.5509	0.1241	2303.4049	10.9925	0.1727	1993.6167	10.9233	0.2696	3626.0724
SALSA	11.8211	0.1640	**166.5997**	12.0385	0.1640	190.7876	10.0407	0.0144	1726.2268
YALL1	10.7026	0.0546	2805.6211	11.2490	0.0765	2995.1133	10.6698	0.0141	10366.2891
ℓ_1-ls	10.6210	0.2696	4745.5160	**9.6899**	0.3688	7597.0326	13.3593	0.5284	5876.1806

in the newly obtained ℓ_0 norm based sparse TFDs. The problem is obvious from comparison of the visually better concentrated sparse TFDs with all the components present (obtained with e.g. AIHT$_K$, ECME$_K$, and DORE$_K$ algorithm) with the visually worse concentrated sparse TFDs with the missing component (obtained with e.g. IHT$_K$, and IHT$_\lambda$ algorithm), as the concentration measure often favours the less concentrated ℓ_0 norm based sparse TFDs with the missing component. On the other hand, execution times of the reconstruction algorithms based on the ℓ_0 norm minimization significantly benefitted from the larger CS-AF area; in only 7 instances execution time of the reconstruction algorithm with the adaptively detected CS-AF sensing mask is slightly larger then the execution time of the reconstruction algorithm with the manually selected CS-AF sensing mask.

The adaptively detected elliptical CS-AF sensing mask, as shown in Table 5.2, brought further improvement of the concentration measure and the Rényi entropy of the resulting ℓ_0 norm based sparse TFDs. However, there are no visually identifiable differences when compared to the ℓ_0 norm based TFDs obtained with the adaptively detected rectangular CS-AF sensing mask, hence the ℓ_0 norm based sparse TFDs with the adaptively detected elliptical CS-AF sensing mask have been omitted from the respective figures. The sparse reconstruction algorithm execution times with the adaptively detected elliptical and rectangular CS-AF area are very similar, some reconstruction algorithms achieved slight acceleration, while other reconstruction algorithms achieved slight deceleration.

In the similar way, as it can be seen by visual inspection of Figs. 5.6 - 5.8, the ℓ_1 norm based sparse TFDs with the adaptively detected CS-AF sensing mask gained significant concentration enhancement when compared to the respective ℓ_1 norm based sparse TFDs with the manually selected CS-AF sensing mask, shown in Figs. 3.6 - 3.8. In fact, for the signal $z_{\mathrm{LFM}}(t)$ some of the newly obtained ℓ_1 norm based sparse TFDs are visually very close to the ideal signal TFD, shown in Fig. 2.1(b). This is confirmed by the improvement of the ℓ_1 based sparse TFDs concentration measures and the Rényi entropies, listed in Table 5.3, of approximately 10%, when compared to the concentration measures of the ℓ_1 based sparse TFDs with the manually selected CS-AF area, listed in Table 3.2. The ℓ_1 norm based reconstruction algorithm execution times were not greatly affected by the CS-AF area enlargement: in approximately half of the instances the reconstruction algorithms achieved slight improvement of their execution times.

Furthermore, with the adaptively detected elliptical CS-AF sensing mask, as apparent from Table 5.4, the concentration measures and the Rényi entropies of the resulting ℓ_1 norm based TFDs achieved approximately 1% additional improvement for the signals $z_{\mathrm{LFM}}(t)$ and $z_{\mathrm{bat}}(t)$, and less then 1% deterioration for the signal $z_{\mathrm{nonLFM}}(t)$. The execution times of the ℓ_1 norm based reconstruction algorithms, on the other hand, did not benefited from the elliptical CS-AF area, as the reported results show slightly slower execution times.

5.2 Sparse Signal Reconstruction Algorithm Based on the FICI Rule

In this section, the new sparse reconstruction algorithm has been proposed based on the FICI rule, previously discussed in Section 4.2.4. In the here-proposed algorithm, the shrinkage operator can be either implemented as hard-thresholding (3.18) for the ℓ_0 norm minimization, or soft-thresholding (3.29) for the ℓ_1 norm minimization. The threshold value is detected adaptively with the FICI rule, while the reconstruction algorithm structure is based on the TwIST algorithm [12], i.e. the algorithm iterative solutions are defined with (3.31). The complete FICI based sparse reconstruction algorithm is defined as follows.

1. In the first step, the solution of previous algorithm iteration is updated as:

$$[\boldsymbol{\varsigma}_z(t,f)]^{[n+1]} = \boldsymbol{\psi}^H \left[\mathbf{A}'_z(\nu,\tau) - \boldsymbol{\psi}\,[\boldsymbol{\vartheta}_z(t,f)]^{[n]}\right] + [\boldsymbol{\vartheta}_z(t,f)]^{[n]}. \tag{5.8}$$

   Note that, the initial algorithm solution, $[\boldsymbol{\varsigma}_z(t,f)]^{[0]}$, can be initialized in different ways; however, the best reported convergence rate and the most accurate solution is obtained with the initialization as: $[\boldsymbol{\varsigma}_z(t,f)]^{[0]} = \boldsymbol{\psi}^H \mathbf{A}'_z(\nu,\tau)$.

2. In the next step, $[\boldsymbol{\varsigma}_z(t,f)]^{[n+1]}$ is vectorized, that is:

$$[\boldsymbol{\varsigma}_{z1D}(i)]^{[n+1]} = \text{vec}\left([\boldsymbol{\varsigma}_z(t,f)]^{[n+1]}\right), \tag{5.9}$$

   followed by calculation of $[\tilde{\boldsymbol{\varsigma}}_{z1D}(i)]^{[n+1]}$, which is performed by sorting the vectorized TFD, $[\boldsymbol{\varsigma}_{z1D}(i)]^{[n+1]}$, in the ascending order, with the zero indexes being removed.

3. The regularization parameter, λ, is obtained as the value of first sample of the sorted vectorized TFD, $[\tilde{\boldsymbol{\varsigma}}_{z1D}(t,f)]^{[n+1]}$, for which the amount of intersection between the respective confidence intervals is bellow the preset threshold value, R_c. The thresholded TFD, $[\tilde{\boldsymbol{\varsigma}}_z(t,f)]^{[n+1]}$, is then calculated by either soft-thresholding or hard-thresholding the TFD, $[\boldsymbol{\varsigma}_z(t,f)]^{[n+1]}$, with the previously calculated threshold value.

4. Finally, the solution of current algorithm iteration, $[\boldsymbol{\vartheta}_z(t,f)]^{[n+1]}$, is calculated by combining $[\tilde{\boldsymbol{\varsigma}}_z(t,f)]^{[n+1]}$ with the previous two solutions, that is:

$$[\boldsymbol{\vartheta}_z(t,f)]^{[n+1]} = (1-\alpha_{\text{TWIST}})\,[\boldsymbol{\vartheta}_z(t,f)]^{[n-1]} + (\alpha_{\text{TWIST}} - \beta_{\text{TWIST}})\,[\boldsymbol{\vartheta}_z(t,f)]^{[n]} + \beta_{\text{TWIST}}\,[\tilde{\boldsymbol{\varsigma}}_z(t,f)]^{[n+1]}, \tag{5.10}$$

5. The algorithm steps 1-4 are repeated until the stopping criterion is satisfied. In the algorithm implementation used in this thesis, the stopping criterion is implemented

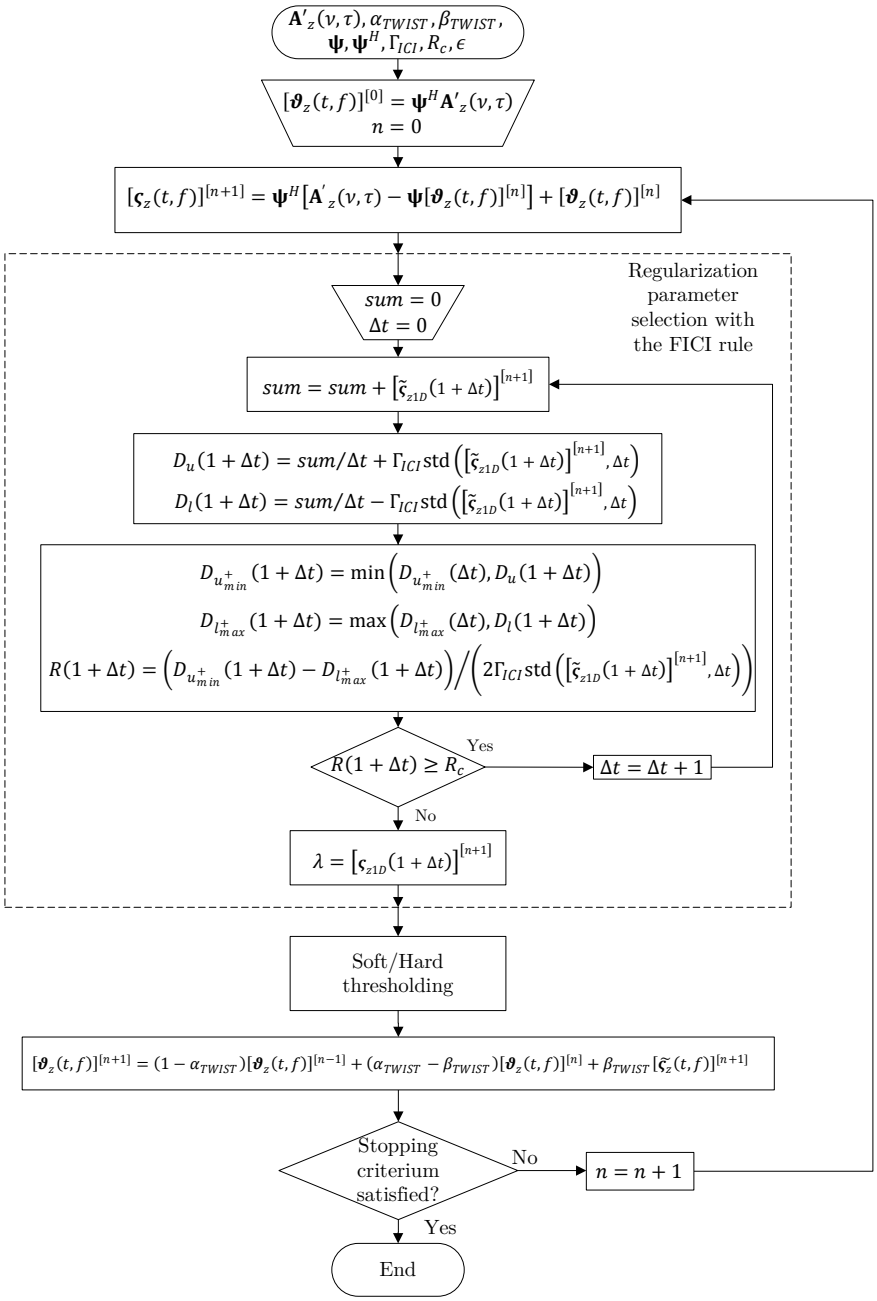


Figure 5.9: Flowchart of the here-proposed sparse reconstruction algorithm based on the FICI rule.

as:

$$\frac{\left|\left|\left|[\boldsymbol{\vartheta}_z(t,f)]^{[n+1]}\right|\right|_2^2 - \left|\left|[\boldsymbol{\vartheta}_z(t,f)]^{[n]}\right|\right|_2^2\right|}{\left|\left|[\boldsymbol{\vartheta}_z(t,f)]^{[n]}\right|\right|_2^2} > \epsilon. \tag{5.11}$$

The flowchart of the complete algorithm is shown in Fig. 5.9, and it is implemented in a MATLAB functions `fun_bpn_FICI_TwIST.m` and `fun_FICIThreshold.m`, the codes of which are given in Appendix C.

Example 5.2. In this example, the sparse TFDs obtained with the reconstruction al-

gorithm based on the FICI rule have been calculated for the previously defined signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$, with three different CS-AF sensing masks: the manually selected CS-AF sensing mask (denoted with the subscript FIX), the adaptively detected rectangular CS-AF sensing mask (denoted with the subscript REC), and the adaptively detected elliptical CS-AF sensing mask (denoted with the subscript ELL). The parameters of the FICI algorithm have been set as: $\Gamma_{\text{ICI}} = 6$, and $R_c = 0.1$, the TwIST relaxation parameters have been set as: $\alpha_{\text{TWIST}} = 1$, and $\beta_{\text{TWIST}} = 0.5$, while the stopping criterium has been set as: $\epsilon = 1\%$. The resulting sparse TFDs based on the ℓ_0 norm minimization are shown in Fig. 5.10, while the resulting sparse TFDs based on the ℓ_1 norm minimization are shown in Fig. 5.11, with the obtained TFD measures summarized in Table 5.5.

By visual inspection of the sparse TFDs obtained by the ℓ_0 norm minimization with the FICI based threshold detection, shown in Fig. 5.10, it can be seen that they are generally better concentrated than the previously obtained ℓ_0 norm based sparse TFDs (shown in Figs. 3.3 - 3.5 and Figs. 5.3 - 5.5). Furthermore, in case of the signals $z_{\text{LFM}}(t)$ and $z_{\text{bat}}(t)$ the weakest signal component is significantly more emphasized, while for the signal $z_{\text{nonLFM}}(t)$, the nonlinear frequency modulated signal component did not get hard-thresholded. On the other hand, the here-proposed FICI based reconstruction algorithm does not require input about the number of non-zero elements in the resulting sparse TFD, eliminating the need of *a priori* information about the signal nature. By comparing the TFD reconstruction results (shown in Table 5.5), with the previously obtained results (shown in Table 3.1, Table 5.1, and Table 5.2) it can be seen that the here-proposed algorithm is among the fastest used reconstruction algorithms. Furthermore, the difference between the concentration measures can be explained, as previously discussed in Example 3.2 and Example 5.1, with the inconsistency in the concentration measure when one of the signal components gets eliminated by the hard-thresholding procedure.

In case when the here-proposed sparse reconstruction algorithm is utilized for the ℓ_1 norm minimization, the resulting sparse TFD concentrations are slightly lower, as evident

Table 5.5: Rényi entropies, $R_z^{\alpha_{\text{R}}}$, concentration measures, M_z^{S}, and the algorithm execution times of the sparse reconstruction algorithm based on the FICI rule. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

		$z_{\text{LFM}}(t)$			$z_{\text{nonLFM}}(t)$			$z_{\text{bat}}(t)$		
		$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{S}}=2$	t $[ms]$	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{S}}=2$	t $[ms]$	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{S}}=2$	t $[ms]$
	Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
ℓ_0 based	FICI_{FIX}	13.7666	0.4231	**27.3776**	13.6805	0.4572	**27.2948**	13.8853	0.2866	**70.4350**
	FICI_{REC}	12.5035	0.2969	27.9221	12.8608	0.3830	27.6586	12.8352	0.2114	71.3774
	FICI_{ELL}	**12.1534**	**0.2586**	28.9959	**12.7241**	**0.3364**	28.7127	**12.5920**	**0.2087**	75.6932
ℓ_1 based	FICI_{FIX}	13.1988	0.2746	**137.1290**	13.1453	0.3351	**110.2280**	13.5192	0.2077	**209.6734**
	FICI_{REC}	11.8151	0.1538	141.6669	12.2025	0.2476	112.2324	12.4146	0.1419	212.6088
	FICI_{ELL}	**11.3254**	**0.0976**	209.5883	**12.0000**	**0.1906**	146.5532	**12.1701**	**0.1301**	228.5196

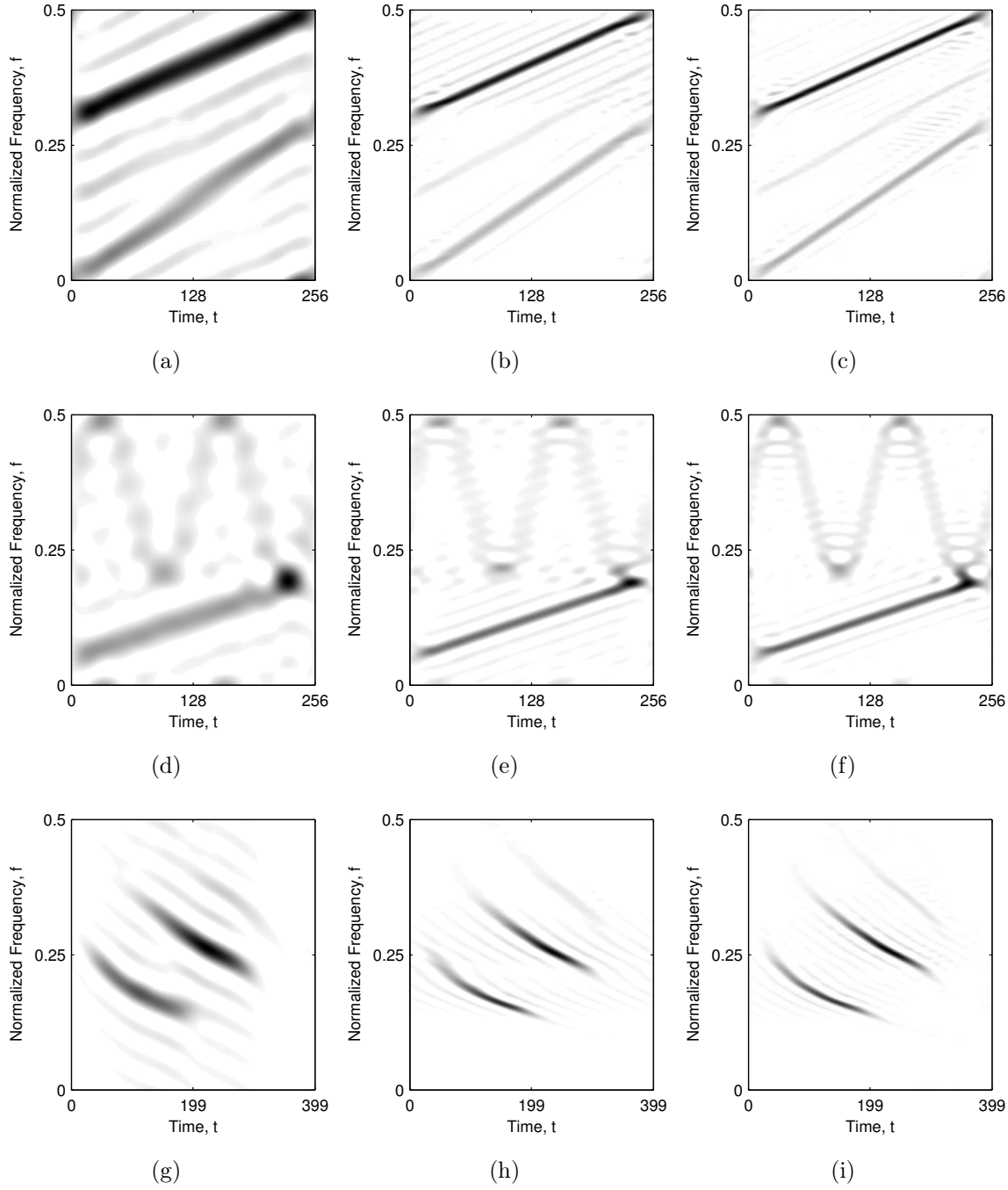


Figure 5.10: The reconstructed sparse TFDs based on the FICI rule and the ℓ_0 norm minimization: (a)-(c) signal $z_{\mathrm{LFM}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively, (d)-(f) signal $z_{\mathrm{nonLFM}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively, (g)-(i) signal $z_{\mathrm{bat}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively.

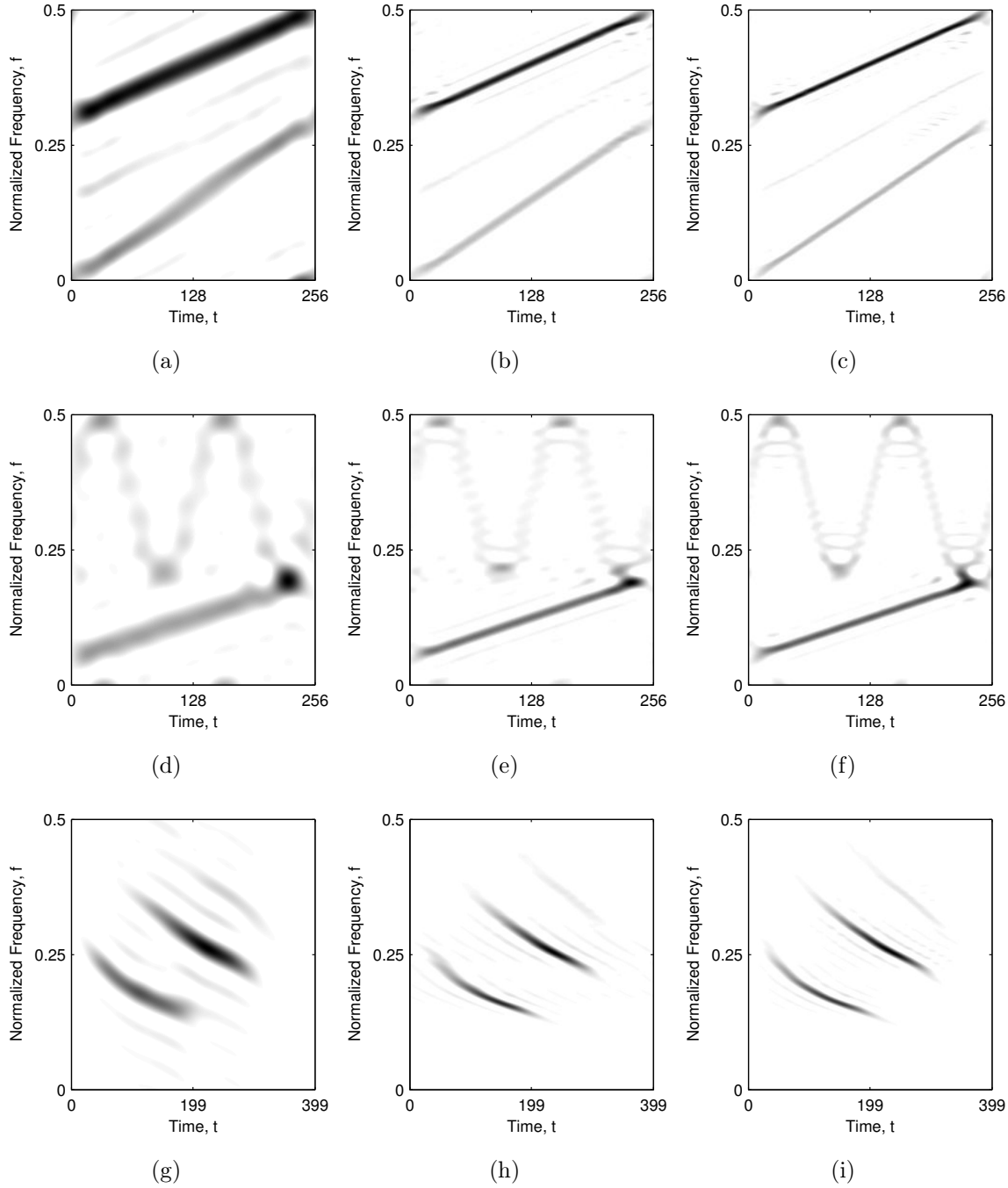


Figure 5.11: The reconstructed sparse TFDs based on the FICI rule and the ℓ_1 norm minimization: (a)-(c) signal $z_{\mathrm{LFM}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively, (d)-(f) signal $z_{\mathrm{nonLFM}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively, (g)-(i) signal $z_{\mathrm{bat}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively.

from the visual comparison of the obtained sparse TFDs (shown in Fig. 5.11), with the previously obtained sparse TFDs based on the ℓ_1 norm minimization (shown in Figs. 3.6 - 3.8 and Figs. 5.6 - 5.8). However, in case of the signal $z_{\text{nonLFM}}(t)$, the nonlinear frequency modulated signal component is more emphasized, while in case of the signal $z_{\text{bat}}(t)$, the extra signal component is revealed. The adaptiveness of the FICI algorithm has influenced the execution time of the ℓ_1 norm based sparse reconstruction algorithm in the same way as it has influenced the execution time of the ℓ_0 norm minimization; the here-proposed algorithm is among the fastest ℓ_1 norm based sparse reconstruction algorithms, as evident from comparison of the obtained results (shown in Table 5.5) with the previously obtained results (shown in Table 3.2, Table 5.3, and Table 5.4). In fact, for the signal $z_{\text{bat}}(t)$, the here-proposed sparse reconstruction algorithm achieves the fastest execution time among all used sparse reconstruction algorithms.

5.3 Sparse Signal Reconstruction Algorithm Based on the Localized Rényi Entropy Information

As already discussed in Section 3.3.3, when a hard-thresholding function is implemented with the operator H_K, the solution of the ℓ_0 norm based sparse reconstruction algorithm is highly dependent on the *a priori* information about the expected sparsity level of the solution, K, that is, in the TFD content, number of signal components, $K \sim N_c N_t$. The algorithm based on the localized Rényi entropy, discussed in Section 4.3, returns the instantaneous number of signal components present in the TFD, $n_c(t)$. This information can be easily used to remove the requirement of *a priori* knowledge about the TFD by adaptively calculating the expected number of non-zero elements in the solution, that is by summing all values of $n_c(t)$:

$$K = K_m \sum_{t=0}^{N_t} n_c(t), \tag{5.12}$$

where K_m is the user defined sparsity relaxation parameter. Without the sparsity relaxation parameter, that is, with $K_m = 1$, the resulting sparse TFDs tend to be oversparse, with some of the weaker components missing. This is because signal components are not perfectly localized, and some of the samples from the stronger signal components can overshadow the samples from the weaker signal components. By setting $K_m > 1$ this effect can be compensated by ensuring more than one sample in the resulting sparse TFD per time slice and signal component. In the conducted experiments it has been shown that setting $5 \leq K_m \leq 10$ is a good choice.

Example 5.3. In this example, the reconstructed sparse TFDs based on the ℓ_0 norm minimization have been calculated for the signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$, with

the reconstruction algorithms which utilize the H_K operator, denoted in the previous examples with the subscript K for three different CS-AF sensing masks: the manually selected CS-AF sensing mask, the adaptively detected rectangular CS-AF sensing mask, and the adaptively detected elliptical CS-AF sensing mask. The simulation set-up is the same as in Example 3.2 and Example 5.1, making the previously obtained results, presented in Table 3.1, Table 5.1, and Table 5.2, comparable with the results obtained in this example, which are summarized in Tables 5.6 - 5.8. Note that the expected sparsity level, K, has been pre-computed, i.e. the time needed to calculate K has been excluded from the algorithm execution times.

In the previous examples the expected sparsity level has been calculated as $K = 5N_tN_c$, where N_c is *a priori* known number of signal components, resulting in $K_{\text{LFM}} = 3840$ samples, $K_{\text{nonLFM}} = 2560$ samples, and $K_{\text{bat}} = 7980$ samples, for the respective signals. On the other hand, the algorithm based on the localized Rényi entropy, with the sparsity relaxation parameter $K_m = 7$, has resulted in a $K_{\text{LFM}} = 3584$ samples, $K_{\text{nonLFM}} = 3360$ samples, and $K_{\text{bat}} = 2527$ samples, for the respective signals. For the signals $z_{\text{LFM}}(t)$, and $z_{\text{nonLFM}}(t)$ the manually and the adaptively calculated sparsity levels are similar, resulting in the visually similar TFDs with the similar values of the Rényi entropies, the concentration measures, and the reconstruction algorithm execution times. On the other hand, for the signal $z_{\text{bat}}(t)$, the difference between the manually and the adaptively calculated sparsity level is over 3 times, which resulted in a general oversparsity of the resulting reconstructed sparse TFDs. This is because, the number of signal components has been deliberately overvalued in the manual selection of the expected sparsity level, in order to compensate the fact that the real-life measured signals do not have to be perfectly localized in frequency.

Table 5.6: Rényi entropies, $R_z^{\alpha_{\text{R}}}$, concentration measures, M_z^{S}, and the algorithm execution times of the reconstruction algorithms based on the ℓ_0 norm minimization with the localized Rényi entropy based sparsity level detection and with the manually selected CS-AF sensing mask. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\text{LFM}}(t)$			$z_{\text{nonLFM}}(t)$			$z_{\text{bat}}(t)$		
	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{s}}=2$	t $[ms]$	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{s}}=2$	t $[ms]$	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{s}}=2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IHT_K	11.7823	0.0547	134.4265	11.6097	0.0510	213.1847	11.2509	0.0158	1015.2998
NIHT_K	11.5935	0.0534	105.1059	10.4917	0.0463	325.7706	**10.8873**	**0.0153**	688.3073
AIHT_K	**10.0438**	0.0463	2781.3772	10.0652	0.0426	2539.2384	11.2786	0.0158	33.8833
ECME_K	10.0657	**0.0453**	969.3566	10.0827	0.0431	939.6820	11.2760	0.0158	61.6094
DORE_K	10.0800	0.0462	2171.7668	**9.8630**	**0.0421**	1853.7792	11.1700	0.0156	93.7271
HTP_K	11.7851	0.0548	21.0307	11.4496	0.0504	20.5785	11.2450	0.0158	70.7071
FHTP_K	14.2817	0.7281	**8.3342**	14.1350	0.7382	**8.2014**	14.5039	0.5680	**30.4408**

Table 5.7: Rényi entropies, $R_z^{\alpha_{\mathrm{R}}}$, concentration measures, M_z^{S}, and the algorithm execution times of the reconstruction algorithms based on the ℓ_0 norm minimization with the localized Rényi entropy based sparsity level detection and with the adaptively detected rectangular CS-AF sensing mask. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{S}}=2$	t $[ms]$	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{S}}=2$	t $[ms]$	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{S}}=2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IHT_K	11.4322	0.0522	85.3947	11.4187	0.0496	11.3064	11.1636	0.0156	284.3883
NIHT_K	11.4171	0.0523	82.0723	11.3800	0.0492	131.8601	**11.0978**	0.0155	414.0112
AIHT_K	11.1567	**0.0489**	2666.3740	10.3757	0.0443	2569.1989	11.1876	0.0157	35.1514
ECME_K	11.3278	0.0510	556.2766	11.2502	0.0474	544.8806	11.1744	0.0156	64.6773
DORE_K	**11.0255**	0.0490	2135.5477	**10.3284**	**0.0437**	2529.3203	11.1107	**0.0154**	95.5262
HTP_K	11.3893	0.0516	21.3178	11.4171	0.0496	20.7766	11.1487	0.0156	77.2793
FHTP_K	13.1762	0.6142	**8.0097**	13.4477	0.6808	**7.9201**	13.5272	0.4687	**31.5556**

Table 5.8: Rényi entropies, $R_z^{\alpha_{\mathrm{R}}}$, concentration measures, M_z^{S}, and the algorithm execution times of the reconstruction algorithms based on the ℓ_0 norm minimization with the localized Rényi entropy based sparsity level detection and with the adaptively detected elliptical CS-AF sensing mask. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\mathrm{LFM}}(t)$			$z_{\mathrm{nonLFM}}(t)$			$z_{\mathrm{bat}}(t)$		
	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{S}}=2$	t $[ms]$	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{S}}=2$	t $[ms]$	$R_z^{\alpha_{\mathrm{R}}}$ $\alpha_{\mathrm{R}}=3$	M_z^{S} $p_{\mathrm{S}}=2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
IHT_K	11.3157	0.0514	34.8242	11.3964	0.0494	34.6914	11.1246	0.0155	289.2149
NIHT_K	11.3299	0.0518	88.4858	11.4194	0.0494	145.6427	**10.9668**	**0.0151**	655.5563
AIHT_K	11.1089	0.0490	1678.3128	10.7800	0.0457	1974.9071	11.1514	0.0156	45.1737
ECME_K	11.1355	**0.0489**	976.8801	11.1584	0.0468	698.0162	11.1377	0.0156	75.6190
DORE_K	**11.0696**	**0.0489**	2144.8454	**10.3658**	**0.0437**	2200.6648	11.0899	0.0154	125.9507
HTP_K	11.3095	0.0513	23.7355	11.3893	0.0494	23.3636	11.1117	0.0155	83.2873
FHTP_K	12.7955	0.5513	**9.9963**	13.3113	0.6202	**9.8692**	13.3016	0.4865	**38.4122**

The calculation of the expected TFD sparsity level, calculated by (5.12), seems wasteful, since the algorithm based on the localized Rényi entropy returns the instantaneous number of components, which is then followed by summation w.r.t. time, in which the previously obtained time dependency is discarded. The instantaneous number of components can be used in order to hard-threshold each time slice of the TFD independently, i.e. with the different value of K, leaving only $K(t) = K_m n_c(t)$ highest valued samples per time slice. The here-proposed sparse reconstruction algorithm is based on the TwIST algorithm [12], and is similar to the FICI based sparse reconstruction algorithm, discussed in Section 5.2, i.e. steps 2 and 3 of the FICI based sparse reconstruction algorithm are replaced with the following step.

2. The hard-thresholded TFD, $[\tilde{\varsigma}_z(t,f)]^{[n+1]}$, is calculated as:

$$[\tilde{\varsigma}_z(t,f)]^{[n+1]} = H_{K(t)}\left\{[\varsigma_z(t,f)]^{[n+1]}\right\}, \tag{5.13}$$

where the operator $H_{K(t)}$ denotes a time-independent hard-thresholding which leaves only the $K(t) = K_m n_c(t)$ highest valued TFD samples per time slice t.

The flowchart of the here-proposed algorithm is shown in Fig. 5.12, and it is implemented in a MATLAB functions `fun_bpn_LocRenyiReconstruction.m` and `fun_CompNumRenyi.m`, the codes of which are given in Appendix D.

Example 5.4. In this example, the sparse TFDs obtained with the ℓ_0 norm based reconstruction algorithm and the time independent hard-thresholding have been calculated for the previously defined signals $z_{\text{LFM}}(t)$, $z_{\text{nonLFM}}(t)$, and $z_{\text{bat}}(t)$. Three different CS-AF sensing masks have been used: the manually selected CS-AF sensing mask (denoted as FIX), the adaptively detected rectangular CS-AF sensing mask (denoted as REC), and the adaptively detected elliptical CS-AF sensing mask (denoted as ELL). The parameters of the algorithm based on the localized Rényi entropy are the same as in Example 4.5, that is, the used instantaneous number of components has already been calculated in Example 4.5, and are shown in Fig. 4.12. The TwIST relaxation parameters have been set as: $\alpha_{\text{TWIST}} = 1$, and $\beta_{\text{TWIST}} = 1$, while the algorithm stopping criterium has been set as: $\epsilon = 0.1\%$. The obtained sparse TFDs are shown in Fig. 5.13, with the reconstruction results summarized in Table 5.9.

By visual comparison of the obtained results (shown in Fig. 5.13), with the previously obtained ℓ_0 norm based sparse TFDs (shown in Figs. 3.3 - 3.5, Figs. 5.3 - 5.5 and Fig. 5.11), it can be concluded that the obtained ℓ_0 norm based sparse TFDs are the best TFDs yet obtained with the ℓ_0 norm minimization, as the auto-terms have the best concentration, and furthermore, non of the components got hard-thresholded in the reconstruction algorithm. This is substantiated with the lowest concentration measures and the Rényi entropies of the resulting sparse TFDs, as it can be seen from comparison of the obtained

Table 5.9: Rényi entropies, $R_z^{\alpha_{\text{R}}}$, concentration measures, M_z^{S}, and the algorithm execution times of the reconstruction algorithms based on the ℓ_0 norm minimization with the time independent thresholding. The bold value indicates the best performing reconstruction algorithm according to the respective measure for the considered signal.

	$z_{\text{LFM}}(t)$			$z_{\text{nonLFM}}(t)$			$z_{\text{bat}}(t)$		
	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{s}}=2$	t $[ms]$	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{s}}=2$	t $[ms]$	$R_z^{\alpha_{\text{R}}}$ $\alpha_{\text{R}}=3$	M_z^{S} $p_{\text{s}}=2$	t $[ms]$
Ideal	9.2568	0.0111	–	8.9944	0.0078	–	–	–	–
FIX	11.6513	0.0500	**201.9226**	**10.1726**	**0.0441**	5076.5932	11.3005	0.0183	809.9706
REC	11.0908	0.0452	1176.7283	10.8688	0.0457	4006.1233	11.2869	0.0181	**428.4814**
ELL	**10.8741**	**0.0441**	1410.4746	10.8283	0.0459	**3216.5214**	**11.1241**	**0.0176**	1139.3701

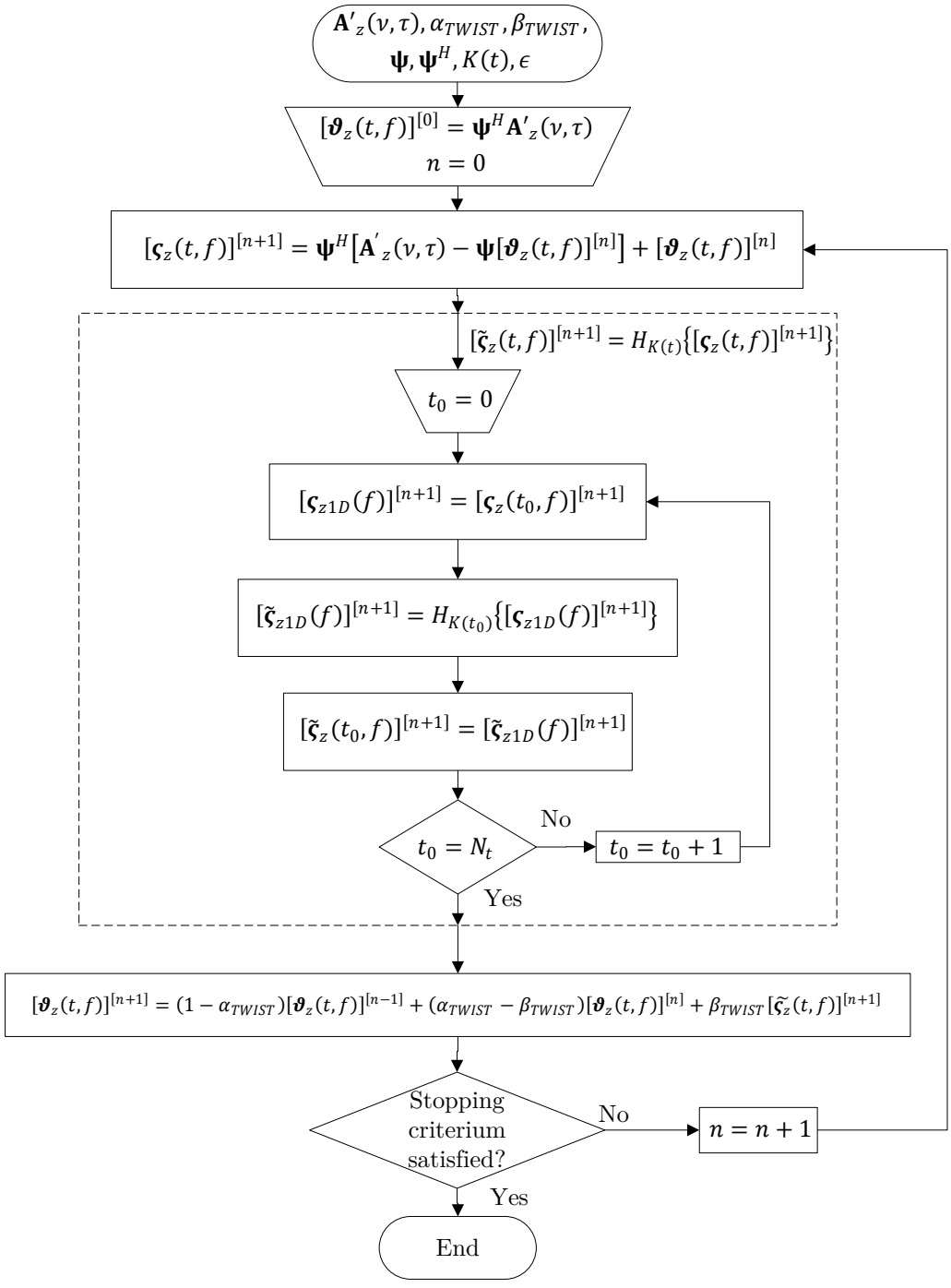


Figure 5.12: Flowchart of the here-proposed sparse reconstruction algorithm based on the ℓ_0 norm minimization with the time independent thresholding.

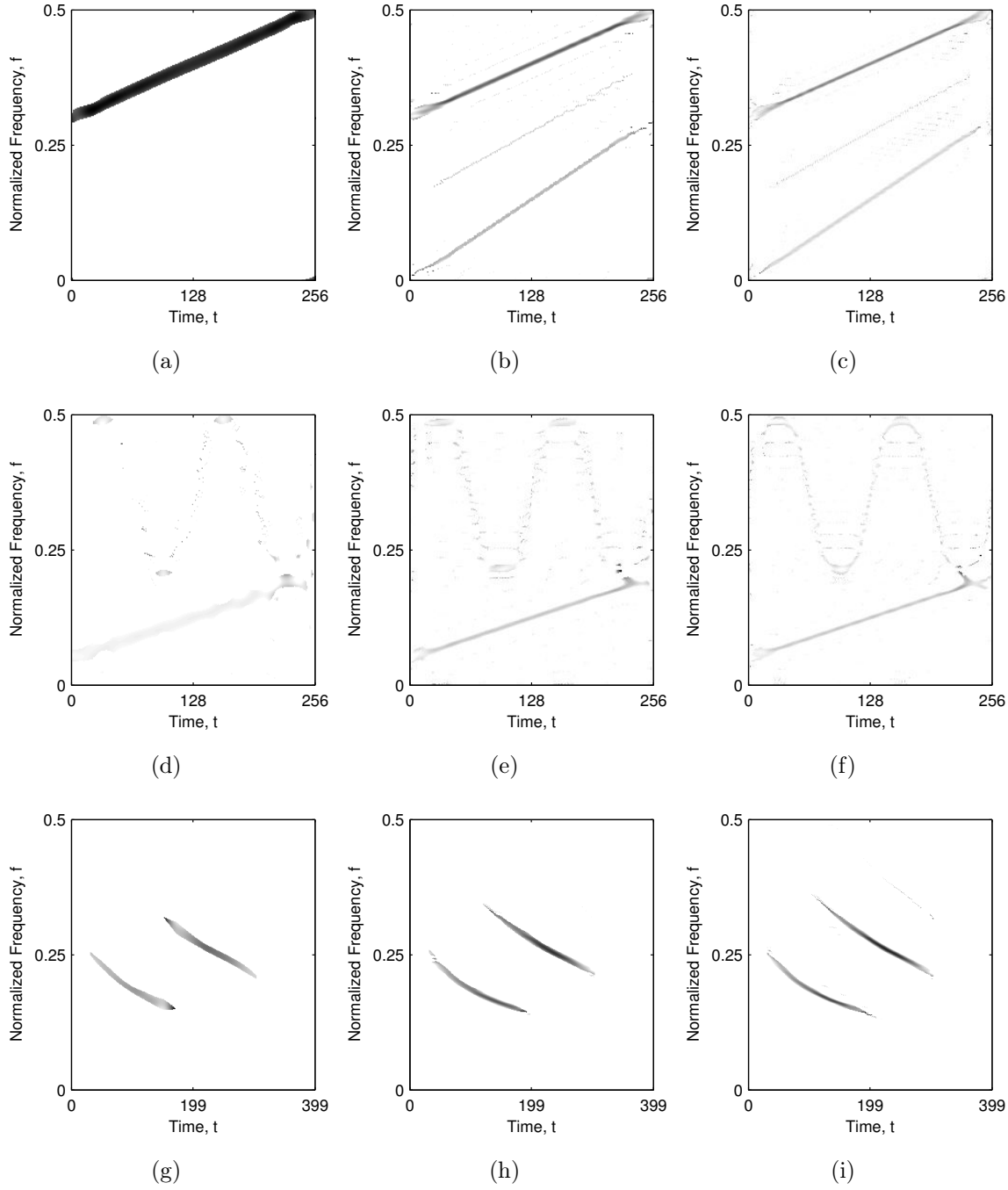


Figure 5.13: The reconstructed sparse TFDs based on the ℓ_0 norm minimization with the time independent thresholding: (a)-(c) signal $z_{\mathrm{LFM}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively, (d)-(f) signal $z_{\mathrm{nonLFM}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively, (g)-(i) signal $z_{\mathrm{bat}}(t)$ with the manually selected, the adaptively detected rectangular, and the adaptively detected elliptical CS-AF sensing mask, respectively.

results (shown in Table 5.9), with the previously obtained results (shown in Table 3.1, Table 5.1, Table 5.2, and Table 5.5), despite the previously discussed inconsistency in the concentration measures when a signal component is missing. However, the described noticeable improvement in the performance of the sparse reconstruction algorithm has costed in the significantly larger algorithm execution times, resulting in the highest measured execution times among the ℓ_0 norm based sparse reconstruction algorithms. This is mainly the case because, before hard-thresholding each time slice of the sparse TFD has to be independently sorted in the ascending order. In addition, the concentration measures achieved in this example are even better than the concentration measures of the ℓ_1 norm based sparse TFDs (shown in Table 3.2, and Tables 5.3 - 5.5), with still higher, but more comparable algorithm execution times.

5.4 Summary

In this chapter, three methods for the concentration enhancement of the signal TFD have been proposed. The first method adaptively detects the largest possible rectangular or elliptical CS-AF sensing mask which ensures the highest possible number of cross-term free samples to be processed by the sparse reconstruction algorithm. In the conducted experiments it has been shown that using the CS-AF sensing mask obtained in the here-proposed way results in the significant concentration enhancement of the resulting sparse TFDs and in the reconstruction algorithm execution time reduction, when compared to the sparse TFD reconstruction using the manually selected CS-AF sensing mask, that is with the CS-AF sensing mask described in the literature. The remaining two here-proposed methods are the adaptive sparse reconstruction algorithms based on the FICI rule and the localized Rényi entropy, respectively. The FICI based sparse reconstruction algorithm can be used for the ℓ_0 norm and the ℓ_1 norm minimization, and in both cases, as demonstrated in the conducted experiments, it has resulted in the significant reduction of the algorithm execution time, when compared to the currently available state-of-the-art sparse reconstruction algorithms. On the other hand, the sparse reconstruction algorithm based on the localized Rényi entropy can be used for the ℓ_0 norm minimization, and in the conducted experiments has resulted in the best concentrated sparse TFDs among the tested sparse reconstruction algorithms. Both of the sparse reconstruction algorithms can be used in conjunction with the here-proposed adaptive CS-AF sensing mask selection method in order to enhance the concentration of the reconstructed sparse TFDs, even further, as confirmed by examples presented in this Chapter.

Chapter 6

Conclusion and Future Work

6.1 Main Conclusions

Time-frequency distributions are powerful set of tools for nonstationary signal analysis, as they provide insight into distribution of the signal energy as a function of both time and frequency, simultaneously. However, the commonly used TFD calculation methods, namely the QTFDs and the higher order TFDs, generate the unwanted artefacts, also called the cross-terms, corrupting the useful auto-terms and making the interpretation of the TFD more difficult. Methods based on the compressive sensing and signal recovery using the sparsity constraints have been utilized in order to gain the highly concentrated sparse TFDs. However, the resulting sparse TFDs are highly dependent on the number of experimentally selected input parameters, thus decreasing the overall viability of the methods in broad application. The goal of this thesis has been to overcome the limitations of the sparsity based methods applied in the time-frequency signal processing, ultimately resulting in a highly concentrated sparse TFDs with the cross-terms being significantly suppressed. The goal is achieved through the adaptive methods proposed in this thesis, and summarized bellow.

The method described in Section 4.1 and Section 5.1 adaptively detects the optimal compressive sensing area in the ambiguity function. The idea behind the proposed method is to capture the biggest rectangular or elliptical compressive sensing area around the AF origin without the cross-terms inclusion. This is achieved by searching the zero-doppler and the zero-lag AF slices for the energy spikes caused by the cross-terms. In the conducted experiments when the CS-AF area is selected with the here-proposed method, the TFD reconstruction based on the ℓ_0 norm and the ℓ_1 norm minimization has resulted in the significantly more concentrated sparse TFDs with the faster reconstruction algorithm convergence rate, when compared to the sparse signal reconstruction using traditionally selected CS-AF area.

The method presented in Section 4.2.4 and Section 5.2 adaptively detects the optimal

thresholding value in the TFD reconstruction process and can be utilized in both ℓ_0 norm and ℓ_1 norm minimization based sparse TFD reconstruction. The method is based on the fast intersection of confidence intervals rule in combination with the TwIST sparse reconstruction algorithm. It has been shown that the here-proposed FICI based sparse reconstruction algorithm is one of the fastest converging algorithms among the currently available state-of-the-art sparse reconstruction algorithms, while keeping the satisfactory TFD concentration level, and eliminating the influence of the experimental threshold value selection on the resulting sparse TFD.

The method introduced in Section 5.3 uses information obtained with the localized Rényi entropy about the instantaneous number of components present in the TFD. This information is used to eliminate the need of *a priori* knowledge about the sparsity level of the resulting TFD in the ℓ_0 norm based sparse TFD reconstruction process. The proposed localized Rényi entropy based sparse reconstruction algorithm has outperformed all of the tested ℓ_0 norm and ℓ_1 norm based reconstruction algorithms in terms of the concentration of the resulting sparse TFDs.

Furthermore, as supported by the results presented in the thesis, combination of the here-proposed CS-AF area selection method with the here-proposed sparse reconstruction algorithms based on the FICI rule and the localized Rényi entropy has led to further improvement of the sparse TFD concentration.

6.2 Future Research Directions

The optimization problem outlined in Section 3.3.1 of this thesis has been solved by the sparse reconstruction algorithms based on the ℓ_0 and the ℓ_1 norm minimization. In the conducted experiments, the ℓ_0 norm based reconstruction algorithms have shown faster convergence rate, while the ℓ_1 norm based reconstruction algorithms have produced more concentrated sparse TFDs. The sparse TFD reconstruction based on the mixed ℓ_0/ℓ_1 norm minimization should be carefully investigated in order to combine the advantages of both approaches. Furthermore, the here-proposed methods could be applied in such framework to obtain highly concentrated sparse TFDs. In addition, the formulated optimization problem can be solved by the various other methods, one such being the orthogonal matching pursuit. Influence of the here-proposed method for the adaptive CS-AF area selection on such optimization procedures should also be investigated.

In this thesis QTFD class has been used as a starting point for the sparse TFD calculation; it would be interesting to explore other classes of the TFDs as a starting points. It is well known that the linear TFDs do not contain cross-terms, and as such, the whole ambiguity function could be used in the reconstruction process. However, the starting linear TFD is significantly less concentrated than the QTFD, imposing a higher requirements on the sparse reconstruction algorithm. On the other hand, higher

order TFDs could be used in order to obtain more concentrated TFDs. Such TFDs, however, imply more cross-terms between the each pair of the auto-terms, resulting in the smaller compressive sensing area in the ambiguity function, which also means higher optimization requirements. The selection of the starting TFD class seems to involve a trade-off between the size of the compressive sensing area in the ambiguity function, and the inherent initial TFD concentration, thus it should be investigated in which way adaptive methods proposed in this thesis correlate with the other classes of TFDs.

In this thesis, performance evaluation of the obtained sparse TFDs has been done by the concentration measure, M_z^{S}, and by the Rényi entropy, $R_z^{\alpha_{\mathrm{R}}}$. Both of the selected measures produce lower values for the more concentrated TFDs. However, the downside of the selected evaluation measures is lack of the lower limit, that is, if all of the samples in the evaluated TFD are set to zero, both measures will be also equal to zero, thus the TFD in question would be the "best" possible TFD according to the selected evaluation methods. This is of course, not the case, since the TFD in question would not provide any information about the considered signal. In general, if some of the TFD auto-term samples are set to zero, the selected performance measures would favour the TFD with the missing information instead of the original TFD. This is evident from the conducted experiments in which the sparse TFD reconstruction process eliminated one of the signal components, resulting in a lower values of the measures, when compared to the sparse TFD containing all of the signal components. This is the reason why in the conducted experiments, the measures of the modeled ideal TFDs have also been given; to serve as a lower limit in interpretation of the obtained results. However, when dealing with the real-life signals, the ideal TFD is not available, thus lower values of evaluation measures could be easily misinterpreted. To avoid such mistakes in the thesis the additional comparison tool has been the subjective visual inspection. Development of the objective evaluation method for the sparse TFDs, which would take into the account all of the above mentioned specific problems when dealing with a sparse TFD, would therefore be an important contribution to the field of sparse time-frequency signal analysis.

Bibliography

[1] P. Addesso, M. Longo, S. Marano, V. Matta, M. Principe, and I. M. Pinto, "Compressed sensing for time-frequency gravitational wave data analysis," *arXiv preprint arXiv:1605.03496*, 2016.

[2] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *Image Processing, IEEE Transactions on*, vol. 19, no. 9, pp. 2345–2356, Sept 2010.

[3] R. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 4, no. 24, pp. 118–121, 2007.

[4] R. G. Baraniuk and D. L. Jones, "Signal-dependent time-frequency analysis using a radially gaussian kernel," *Signal processing*, vol. 32, no. 3, pp. 263–284, 1993.

[5] ——, "A signal-dependent time-frequency representation: optimal kernel design," *IEEE Transactions on signal processing*, vol. 41, no. 4, pp. 1589–1602, 1993.

[6] R. G. Baraniuk, P. Flandrin, A. J. Janssen, and O. J. Michel, "Measuring time-frequency information content using the rényi entropies," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1391–1409, 2001.

[7] O. Barkan, J. Weill, A. Averbuch, and S. Dekel, "Adaptive compressed tomography sensing," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, June 2013, pp. 2195–2202.

[8] B. Barkat and B. Boashash, "A high-resolution quadratic time-frequency distribution for multicomponent signals analysis," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2232–2239, 2001.

[9] H. Barkhuijsen, R. De Beer, W. Bovee, J. Creyghton, and D. Van Ormondt, "Application of linear prediction and singular value decomposition (LPSVD) to determine NMR frequencies and intensities from the FID," *Magnetic resonance in medicine*, vol. 2, no. 1, pp. 86–89, 1985.

[10] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[11] S. Becker, J. Bobin, and E. J. Candès, "NESTA: a fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[12] J. M. Bioucas-Dias and M. A. Figueiredo, "A new TwIST: two-step iterative shrinkage/thresholding algorithms for image restoration," *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2992–3004, 2007.

[13] T. Blumensath, "Accelerated iterative hard thresholding," *Signal Processing*, vol. 92, no. 3, pp. 752–756, 2012.

[14] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[15] B. Boashash, *Time-Frequency Signal Analysis and Processing, 2nd Edition A Comprehensive Reference*. Elsevier, 2016.

[16] B. Boashash and T. Ben-Jabeur, "Design of a high-resolution separable-kernel quadratic TFD for improving newborn health outcomes using fetal movement detection," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*. IEEE, 2012, pp. 354–359.

[17] B. Boashash and V. Sučić, "Resolution measure criteria for the objective assessment of the performance of quadratic time-frequency distributions," *IEEE Transactions on Signal Processing*, vol. 51, no. 5, pp. 1253–1263, 2003.

[18] P. Borgnat and P. Flandrin, "Time-frequency localization from sparsity constraints," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2008, pp. 3785–3788.

[19] T. T. Cai and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4680–4688, July 2011.

[20] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, March 2008.

[21] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[22] H. I. Choi and W. J. Williams, "Improved time-frequency representation of multicomponent signals using exponential kernels," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 6, pp. 862–871, 1989.

[23] J. F. Claerbout and F. Muir, "Robust modeling with erratic data," *Geophysics*, vol. 38, no. 5, pp. 826–844, 1973.

[24] L. Cohen, *Time-Frequency Analysis: Theory and Applications*. Prentice-Hall, 1995.

[25] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on pure and applied mathematics*, vol. 57, pp. 1416–1457, 2004.

[26] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 6–18, Jan 2006.

[27] D. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[28] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. E. Kelly, R. G. Baraniuk *et al.*, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[29] M. A. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 586–597, 2007.

[30] P. Flandrin and P. Borgnat, "Time-frequency energy distributions meet compressed sensing," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 2974–2982, June 2010.

[31] P. Flandrin, N. Pustelnik, and P. Borgnat, "On wigner-based sparse time-frequency distributions," in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Dec 2015, pp. 65–68.

[32] S. Foucart, "Hard thresholding pursuit: an algorithm for compressive sensing," *SIAM Journal on Numerical Analysis*, vol. 49, no. 6, pp. 2543–2563, 2011.

[33] A. Gholami, "Sparse time-frequency decomposition and some applications," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 6, pp. 3598–3604, June 2013.

[34] A. Goldenshluger and A. Nemirovski, "On spatially adaptive estimation of nonparametric regression," *Mathematical methods of Statistics*, vol. 6, no. 2, pp. 135–170, 1997.

[35] A. Gramfort, D. Strohmeier, J. Haueisen, M. Hämäläinen, and M. Kowalski, "Time-frequency mixed-norm estimates: Sparse M/EEG imaging with non-stationary source activations," *NeuroImage*, vol. 70, pp. 410 – 422, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811912012372

[36] E. T. Hale, W. Yin, and Y. Zhang, "A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing," *CAAM TR07-07, Rice University*, vol. 43, pp. 1–44, 2007.

[37] J. A. Högbom, "Aperture synthesis with a non-regular distribution of interferometer baselines," *Astronomy and Astrophysics Supplement Series*, vol. 15, pp. 417–426, 1974.

[38] H. Huang, S. Misra, W. Tang, H. Barani, and H. Al-Azzawi, "Applications of compressed sensing in communications networks," *arXiv preprint arXiv:1305.3002*, 2013.

[39] Z. M. Hussain and B. Boashash, "Adaptive instantaneous frequency estimation of multicomponent FM signals using quadratic time-frequency distributions," *IEEE Transactions on Signal Processing*, vol. 50, no. 8, pp. 1866–1876, 2002.

[40] D. L. Jones and T. W. Parks, "A high resolution data-adaptive time-frequency representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 12, pp. 2127–2135, 1990.

[41] V. Katkovnik, "A new method for varying adaptive bandwidth selection," *IEEE Transactions on signal processing*, vol. 47, no. 9, pp. 2567–2571, 1999. [Online]. Available: http://dx.doi.org/10.1109/78.782208

[42] V. Katkovnik, K. Egiazarian, and J. Astola, *Adaptive Varying Scale Methods in Image Processing*. Tampere: Tampere International Center for Signal Processing, 2003.

[43] ——, *Local approximation techniques in signal and image processing*. SPIE, 2006. [Online]. Available: http://dx.doi.org/10.1117/3.660178

[44] K. Koh, S.-J. Kim, and S. Boyd, "An interior-point method for large-scale l1-regularized logistic regression," *Journal of Machine learning research*, vol. 8, no. Jul, pp. 1519–1555, 2007.

[45] S. Leier and A. M. Zoubir, "Aperture undersampling using compressive sensing for synthetic aperture stripmap imaging," *EURASIP Journal on Advances in*

Signal Processing, vol. 2014, no. 1, pp. 156–170, 2014. [Online]. Available: http://dx.doi.org/10.1186/1687-6180-2014-156

[46] J. Lerga, M. Vrankić, and V. Sučić, "A signal denoising method based on the improved ICI rule," *IEEE Signal Processing Letters*, vol. 15, pp. 601 – 604, 2008. [Online]. Available: http://dx.doi.org/10.1109/LSP.2008.2001817

[47] N. Levanon and E. Mozeson, *Radar Signals*. John Wiley & Sons, 2004.

[48] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, March 2008.

[49] P. J. M. Born, "Zur quantenmechanik," *Zeitschrift fur Physik*, vol. 34, pp. 858–888, 1925.

[50] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec 1993.

[51] H. Margenau and R. N. Hill, "Correlation between measurements in quantum theory," *Progress of Theoretical Physics*, vol. 26, no. 5, pp. 722–738, 1961.

[52] D. Needell and J. Tropp, "CoSaMP: iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301 – 321, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1063520308000638

[53] Y. Oike and A. E. Gamal, "CMOS image sensor with per-column $\Sigma\Delta$ ADC and programmable compressed sensing," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 318–328, Jan 2013.

[54] I. Orović, S. Stanković, and T. Thayaparan, "Time-frequency-based instantaneous frequency estimation of sparse signals from incomplete set of samples," *IET Signal Processing*, vol. 8, no. 3, pp. 239–245, May 2014.

[55] C. H. Page, "Instantaneous power spectra," *Journal of Applied Physics*, vol. 23, no. 1, pp. 103–106, 1952.

[56] V. M. Patel, G. R. Easley, D. M. H. Jr., and R. Chellappa, "Compressed synthetic aperture radar," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 244–254, April 2010.

[57] R. Prony, "Essai experimental et analytique: sur les lois de la dilatabilite de fluides elastique et sur celles de la force expansive de la vapeur de l'alkool, a differentes temperatures," *Journal de l'Ecole Polytechnique Floreal et Plairial*, vol. 1, no. 22, pp. 24–76, 1795.

[58] K. Qiu and A. Dogandžić, "ECME thresholding methods for sparse signal reconstruction," *arXiv preprint arXiv:1004.4880*, 2010.

[59] ——, "Sparse signal reconstruction via ECME hard thresholding," *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4551–4569, 2012.

[60] A. Rihaczek, "Signal energy distribution in time and frequency," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 369–374, 1968.

[61] J. Romberg, "Imaging via compressive sampling [introduction to compressive sampling and recovery via convex programming]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, 2008.

[62] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259 – 268, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016727899290242F

[63] K. Sasaki, S. Kawata, and S. Minami, "Constrained nonlinear method for estimating component spectra from multicomponent mixtures," *Applied Optics*, vol. 22, no. 22, pp. 3599–3603, 1983.

[64] L. Stanković, I. Orović, S. Stanković, and M. Amin, "Compressive sensing based separation of nonstationary and stationary signals overlapping in time-frequency," *Signal Processing, IEEE Transactions on*, vol. 61, no. 18, pp. 4562–4572, Sept 2013.

[65] L. Stanković, S. Stanković, I. Orović, and M. G. Amin, "Robust time-frequency analysis based on the L-estimation and compressive sensing," *IEEE Signal Processing Letters*, vol. 20, no. 5, pp. 499–502, 2013.

[66] L. Stanković, S. Stanković, T. Thayaparan, M. Daković, and I. Orović, "Separation and reconstruction of the rigid body and micro-doppler signal in ISAR part I - theory," *IET Radar, Sonar Navigation*, vol. 9, no. 9, pp. 1147–1154, 2015.

[67] L. Stanković, S. Stanković, I. Djurović, and M. Daković, Eds., *Time-Frequency Signal Analysis*, 2011.

[68] L. Stanković, "A measure of some time-frequency distributions concentration," *Signal Processing*, vol. 81, no. 3, pp. 621–631, 2001.

[69] S. Stanković, I. Orović, and M. Amin, "L-statistics based modification of reconstruction algorithms for compressive sensing in the presence of impulse noise," *Signal Processing*, vol. 93, no. 11, pp. 2927–2931, 2013.

[70] A. S. Stern and J. C. Hoch, "A new approach to compressed sensing for NMR," *Magnetic Resonance in Chemistry*, vol. 53, no. 11, pp. 908–912, 2015.

[71] V. Sučić, N. Saulig, and B. Boashash, "Estimating the number of components of a multicomponent nonstationary signal using the short-term time-frequency rényi entropy," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, pp. 125–136, 2011. [Online]. Available: http://dx.doi.org/10.1186/1687-6180-2011-125

[72] ——, "Analysis of local time-frequency entropy features for nonstationary signal components time supports detection," *Digital Signal Processing*, vol. 34, pp. 56 – 66, 2014. [Online]. Available: //www.sciencedirect.com/science/article/pii/S1051200414002292

[73] H. L. Taylor, S. C. Banks, and J. F. McCoy, "Deconvolution with the ℓ_1 norm," *Geophysics*, vol. 44, no. 1, pp. 39–52, 1979.

[74] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[75] S. S. Vasanawala, M. T. Alley, B. A. Hargreaves, R. A. Barth, J. M. Pauly, and M. Lustig, "Improved pediatric MR imaging with compressed sensing," *Radiology*, vol. 256, no. 2, pp. 607–616, 2010.

[76] J. Ville, "Theorie et applications de la notion de signal analytique," *Cables Transmission*, vol. 2, pp. 61–74, 1948.

[77] I. Volarić and V. Sučić, "On the noise impact in the l1 based reconstruction of the sparse time-frequency distributions," in *2016 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*, Sept 2016, pp. 1–6.

[78] I. Volarić, V. Sučić, and Z. Car, "A compressive sensing based method for cross-terms suppression in the time-frequency plane," in *2015 IEEE 15th International Conference on Bioinformatics and Bioengineering (BIBE)*, Nov 2015, pp. 1–4.

[79] I. Volarić, J. Lerga, and V. Sučić, "A fast signal denoising algorithm based on the LPA-ICI method for real-time applications," *Circuits, Systems, and Signal Processing*, pp. 1–17, 2017. [Online]. Available: http://dx.doi.org/10.1007/s00034-017-0538-1

[80] E. Wigner, "On the quantum correction for thermodynamic equilibrium," *Physical review*, vol. 40, no. 5, pp. 749–759, 1932.

[81] W. J. Williams and T. Sang, "Adaptive RID kernels which minimize time-frequency uncertainty," in *Time-Frequency and Time-Scale Analysis, 1994., Proceedings of the IEEE-SP International Symposium on.* IEEE, 1994, pp. 96–99.

[82] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, "Sparse reconstruction by separable approximation," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2479–2493, 2009.

[83] J. Yang and Y. Zhang, "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM journal on scientific computing*, vol. 33, no. 1, pp. 250–278, 2011.

[84] T. Zhang, "Sparse recovery with orthogonal matching pursuit under RIP," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6215–6221, Sept 2011.

[85] Y. Zhang, "Users guide for YALL1: Your ALgorithms for l1 optimization," *Technique report*, pp. 09–17, 2009.

List of Figures

List of Tables

List of Symbols

Latin letters:

$[\cdot]^{[n]}$	-	solution of the n-th algorithm iteration
$a_{1_{\text{DORE}}}$, $a_{2_{\text{DORE}}}$	-	line search parameters of the double overrelaxation thresholding scheme algorithm
$A_i(t)$	-	amplitude of the i-th component in $s(t)$
$A_z(\nu,\tau)$	-	ambiguity function of $z(t)$
$\mathcal{A}_z(\nu,\tau)$	-	filtered ambiguity function of $z(t)$
$\mathbf{A}_z(\nu,\tau)$	-	matrix representation of the $A_z(\nu,\tau)$
$\mathbf{A}'_z(\nu,\tau)$	-	compressive sensed $\mathbf{A}_z(\nu,\tau)$ (i.e. observation or measurement matrix)
$c(\boldsymbol{\vartheta}_z(t,f))$	-	regularization function (in general)
$c^{\ell_0}(\boldsymbol{\vartheta}_z(t,f))$	-	ℓ_0 norm based $c(\boldsymbol{\vartheta}_z(t,f))$
$c^{\ell_1}(\boldsymbol{\vartheta}_z(t,f))$	-	ℓ_1 norm based $c(\boldsymbol{\vartheta}_z(t,f))$
$c^{\ell_2}(\boldsymbol{\vartheta}_z(t,f))$	-	ℓ_2 norm based $c(\boldsymbol{\vartheta}_z(t,f))$
$D(t_0+\Delta t)$	-	confidence interval length
$D_{\text{l}}(t_0+\Delta t)$	-	lower boundary of the confidence interval
$D_{\text{l}^+_{\text{max}}}(t_0+\Delta t)$	-	maximum of $D_{\text{l}}(t_0+\Delta t)$
$D_{\text{u}}(t_0+\Delta t)$	-	upper boundary of the confidence interval
$D_{\text{u}^+_{\text{min}}}(t_0+\Delta t)$	-	minimum of $D_{\text{u}}(t_0+\Delta t)$
$e(t,w)$	-	estimation error
E_z	-	energy of signal $z(t)$
$f_{0_i}(t)$	-	instantaneous frequency of the i-th component in $s(t)$
$f\left(\boldsymbol{\vartheta}_z(t,f)\right)$	-	error estimating function (objective function of the minimization)
$f_\lambda\left(\boldsymbol{\vartheta}_z(t,f)\right)$	-	Huber function
T_s, f_s	-	sampling period and frequency, respectfully
$g(\nu,\tau)$	-	kernel in the ambiguity domain
$G(t,\tau)$	-	kernel in the time-lag domain
$\mathcal{G}(\nu,f)$	-	kernel in the doppler-frequency domain
$I(t_0+\Delta t)$	-	amount of intersection between the confidence intervals

K	-	sparsity level
K_m	-	sparsity relaxation parameter
k_{APK}	-	parameter of the adaptive parallelogram 1/0 kernel
M	-	number of Monte Carlo simulations
M_z^{JP}	-	concentration measure proposed by Jones-Park
M_z^{JPL}	-	localized concentration measure proposed by Jones-Park
M_z^{S}	-	concentration measure proposed by Stanković
N_c	-	number of nonstationary components in $s(t)$
$N_{\text{CI}}^{\text{ICI}}$	-	number of calculated confidence intervals by the ICI rule
$N_{\text{CI}}^{\text{FICI}}$	-	number of calculated confidence intervals by the FICI rule
N_{CS}	-	number of compressed sensed samples
N_f	-	number of available time samples
N_f	-	number of available frequency bins
N_ν	-	number of available doppler bins
N'_ν	-	number of lag samples in $\boldsymbol{\phi}(\nu,\tau)$
N_τ	-	number of available lag samples
N'_τ	-	number of doppler bins in $\boldsymbol{\phi}(\nu,\tau)$
$n_c(t)$	-	instantaneous number of signal components
p_{S}	-	parameter of the M_z^{S}
$r(t,w)$	-	point-wise mean square risk
$r^+(t,w)$	-	$r(t,w)$ minimization result over $w(t)$
$r_z(\nu,f)$	-	spectral autocorrelation function of $z(t)$
r_ν, r_τ	-	radiuses of the elliptical CS-AF mask
$R(t_0+\Delta t)$	-	relative amount of confidence interval intersection
R_c	-	threshold value of the RICI rule
$R_z(t,\tau)$	-	localized autocorrelation function of $z(t)$
$R_z^{\alpha_{\text{R}}}$	-	Rényi entropy of order α_{R}
$s(t)$	-	real signal
$\hat{s}(t,w)$	-	estimation of $s(t)$
$\hat{s}^{[m]}(t)$	-	estimation of $s(t)$ in the m-th simulation
$S(f)$	-	Fourier transform of $s(t)$
$S_\tau(\tau)$, $S_\nu(\nu)$	-	zero lag and zero doppler slices of $A_z(\nu,\tau)$
$S_\tau^*(\tau)$, $S_\nu^*(\nu)$	-	filtered $S_\tau(\tau)$ and $S_\nu(\nu)$, respectfully
t_0	-	considered time sample
$\text{SPEC}_z(t,f)$	-	spectrogram of $z(t)$
$\text{STFT}_z(t,f)$	-	short-time Fourier transformation of $z(t)$
$z(t)$	-	analytical associate of $s(t)$
$Z(f)$	-	Fourier transformation of $z(t)$
$w(\tau)$	-	window function

$w^{+}(t)$	-	optimal $w(\tau)$ (obtained with the intersection of confidence intervals rule)
$w^{*}(t)$	-	optimal $w(\tau)$ (obtained with the relative intersection of confidence intervals rule
$\mathbf{W}(\nu, t)$	-	matrix representation of the Fourier transformation
$\mathbf{W}_i(\tau, f)$	-	matrix representation of the inverse Fourier transformation
$x(t)$	-	noise corrupted $s(t)$, i.e. $x(t) = s(t) + \epsilon(t)$
$W_z(t, f)$	-	Wigner-Ville distribution of $z(t)$

Greek letters:

α_{BJ}	-	kernel parameter of the Born-Jordan distribution
α_{TWIST}, β_{TWIST}	-	relaxation parameters of the two step iterative shrinkage/thresholding algorithm
α_{LS}, β_{LS}, μ_{LS}, k_{LS}, t_{LS}	-	parameters of the ℓ_1-ls algorithm
α_{NESTA}, τ_{NESTA}	-	parameters of the Nestrov algorithm
α_{R}	-	order of the Rényi entropy
α_{RGK}	-	volume parameter of the radially Gaussian kernel
α_{SINC}	-	kernel parameter of the Sinc distribution
α_{SPARSA}, η_{SPARSA}	-	parameters of the sparse reconstruction by separable approximation algorithm
α_{EMB}, β_{EMB}	-	kernel parameters of the extended modified B distribution
α_{YALL}, β_{YALL}, γ_{YALL}, μ_{YALL}, ρ_{YALL}	-	parameters of the your augmented Lagrangian algorithm for ℓ_1
β_{B}	-	kernel parameter of the B distribution
β_{IST}	-	relaxation parameter of the iterative shrinkage/thresholding algorithm
β_{MB}	-	kernel parameter of the modified B distribution
β_{FPC}, γ_{FPC}, μ_{FPC}, τ_{FPC}	-	parameters of the fast iterative shrinkage/thresholding algorithm
$\Gamma^{[n]}$	-	support of $\left[\boldsymbol{\vartheta}_z^{\ell_0}(t, f)\right]^{[n]}$
Γ_{ICI}	-	threshold value of the intersection of confidence intervals rule
$\sigma(\psi)$	-	spread function of the radially Gaussian kernel
σ_{CW}	-	kernel parameter of the Choi-Williams distribution
$\boldsymbol{\varsigma}_z(t, f)$	-	auxiliary variable in calculation of $\boldsymbol{\vartheta}_z(t, f)$
$\gamma(t, f)$	-	kernel in the time-frequency domain

Δt	-	window length
$\Delta t(t_0)$	-	window length at t_0
$\Delta t^{+}(t_0)$	-	optimal $\Delta t(t_0)$ obtained with the ICI rule
$\Delta t^{*}(t_0)$	-	optimal $\Delta t(t_0)$ obtained with the RICI rule
$\Delta t_{\mathrm{b}}^{+}(t_0)$	-	optimal backward $\Delta t(t_0)$ obtained with the ICI rule
$\Delta t_{\mathrm{b}}^{*}(t_0)$	-	optimal backward $\Delta t(t_0)$ obtained with the RICI rule
$\Delta t_{\mathrm{f}}^{+}(t_0)$	-	optimal forward $\Delta t(t_0)$ obtained with the ICI rule
$\Delta t_{\mathrm{f}}^{*}(t_0)$	-	optimal forward $\Delta t(t_0)$ obtained with the RICI rule
ϵ	-	accuracy of the $\boldsymbol{\vartheta}_z(t,f)$ reconstruction
ϵ_{APK}	-	threshold value of the adaptive parallelogram $1/0$ kernel
$\epsilon(t)$	-	additive white Gaussian noise
$\zeta(t,w)$	-	zero mean Gaussian random error
η_{ICI}	-	ratio of $e(t,w^{+})$ bias and $\hat{s}(t,w^{+})$ standard deviation
$\boldsymbol{\vartheta}_z(t,f)$	-	matrix representation of the sparse $W_z(t,f)$
$\widehat{\boldsymbol{\vartheta}}_z(t,f)$	-	optimal $\boldsymbol{\vartheta}_z(t,f)$ (in general)
$\boldsymbol{\vartheta}_z^{\ell_0}(t,f)$	-	$\widehat{\boldsymbol{\vartheta}}_z(t,f)$ obtained with the ℓ_0 norm minimization
$\boldsymbol{\vartheta}_z^{\ell_1}(t,f)$	-	$\widehat{\boldsymbol{\vartheta}}_z(t,f)$ obtained with the ℓ_1 norm minimization
$\boldsymbol{\vartheta}_z^{\ell_2}(t,f)$	-	$\widehat{\boldsymbol{\vartheta}}_z(t,f)$ obtained with the ℓ_2 norm minimization
λ	-	regularization parameter
λ_{ϵ}	-	optimal Lagrange multiplier
$\mu(\mathbf{x},\mathbf{y})$	-	mutual coherence between matrices $\mathbf{x}$ and $\mathbf{y}$
μ_{IHT}, μ_{NIHT}, μ_{ECME}	-	step size of the respectful ℓ_0 norm based algorithm
$\hat{\rho}_z(t,f)$	-	ideal time-frequency distribution of $z(t)$
$\rho_z(t,f)$	-	filtered Wigner-Ville distribution of $z(t)$
$\rho_{z_N}(t,f)$	-	$\rho_z(t,f)$ normalized over E_z
$\rho_{z_{t_0}}(t,f)$	-	$\rho_z(t,f)$ localized around t_0
$\phi_t\left(\boldsymbol{\varsigma}_z(t,f),\boldsymbol{\vartheta}_z(t,f)\right)$	-	$f\left(\boldsymbol{\vartheta}_z(t,f)\right)$ updated with the logarithmic barrier $-\boldsymbol{\varsigma}_z(t,f) \leq \boldsymbol{\vartheta}_z(t,f) \leq \boldsymbol{\varsigma}_z(t,f)$
$\varphi_i(t)$	-	phase content of the i-th component in $s(t)$
$\boldsymbol{\phi}(\nu,\tau)$	-	sensing matrix
$\chi_{1-\alpha/2}$	-	$(1-\alpha/2)$th quantile of the standard Gaussian distribution
$\boldsymbol{\psi}$	-	domain transformation matrix (i.e. 2D Fourier transformation)
$\boldsymbol{\psi}_{\Gamma^{[n]}}$	-	$\boldsymbol{\psi}$ with elements outside $\Gamma^{[n]}$ removed

List of Abbreviations

ADMM	-	alternating direction method of multipliers
ADORE	-	automatic double overrelaxation thresholding scheme
AF	-	ambiguity function
AIHT	-	accelerated iterative hard thresholding
AL	-	augmented Lagrangian
APK	-	adaptive parallelogram 1/0 kernel
AWGN	-	additive white Gaussian noise
BB	-	Barzilai-Borwein
CS	-	compressive sensing
CS-AF	-	compressive sensed ambiguity function
CT	-	computed tomography
DORE	-	double overrelaxation thresholding scheme
ECME	-	expectation-conditional maximisation either
FHTP	-	fast hard thresholding pursuit
FICI	-	fast intersection of confidence intervals
FISTA	-	fast iterative shrinkage/thresholding algorithm
FPC	-	fixed point continuation
GPSR	-	gradient projection for sparse reconstruction
HTP	-	hard thresholding pursuit
IHT	-	iterative hard thresholding
ICI	-	intersection of confidence intervals
IST	-	iterative shrinkage/thresholding
LACF	-	localized autocorrelation function
LASSO	-	least absolute shrinkage and selection operator
LFM	-	linear frequency modulated
LPA	-	local polynomial approximation
MAE	-	mean absolute error
MAX	-	maximum absolute difference
MRI	-	magnetic resonance imaging
MSE	-	mean square error

NESTA	-	Nestrov algorithm
NHTP	-	normalized hard thresholding pursuit
NIHT	-	normalized iterative hard thresholding
NMR	-	nuclear magnetic resonance spectroscopy
PCG	-	precondition conjugate gradient
PDF	-	probability density function
QTFD	-	quadratic time-frequency distribution
RGK	-	radial Gaussian kernel
RICI	-	relative intersection of confidence intervals
RIP	-	restricted isometry property
RMSE	-	root of mean square error
SAF	-	spectral autocorrelation function
SALSA	-	split augmented Lagrangian shrinkage algorithm
SpaRSA	-	sparse reconstruction by separable approximation
STFT	-	short-time Fourier transformation
TF	-	time-frequency
TFD	-	time-frequency distribution
TwIST	-	two-step iterative shrinkage/thresholding
USS	-	unconstrained sparsity selection
WVD	-	Wigner-Ville distribution
YALL1	-	your augmented Lagrangian algorithm for ℓ_1

Appendices

Appendix A

MATLAB Code for Multiplication with the Domain Transformation Matrices, ψ and ψ^H

```
function tfd = fun_ambi2tf(amb, N, Ntf, Ntf_fin, amb_mask)
%function tfd = fun_ambi2tf(amb, N, Ntf, Ntf_fin, amb_mask)
%Transformation from the ambiguity function to the time-frequency domain.
%
% Inputs:
%   amb     -  Required, 2D Array. Ambiguity function (AF).
%   N       -  Optional, integer or a 2 element array [Nx, Ny]. Number
%              of points around the AF origin (N <= size(amb)) to take
%              into the account (default: N = size(amb))).
%              Nx - number of doppler bins. Ny - number of lag instances.
%   Ntf     -  Optional, integer or a 2 element array [Ntfx, Ntfy].
%              Number of points in the TF domain (Ntf >= N). Before the
%              transformation AF domain is zero-padded (default: Ntf = N)
%              Ntfx - number of frequency bins. Ntfy - number of time
%              instances.
%   Ntf_fin -  Optional, integer or a 2 element array
%              [Ntfx_fin, Ntfy_fin]. The final number of TFD points.
%              If Ntf_fin > Ntf TFD is zero-padded to the Ntf_fin. If
%              Ntf_fin < Ntf TFD is cut to Ntf_fin
%              (default: Ntf_fin = Ntf).
%              Ntfx_fin - number of frequency bins.
%              Ntfy_fin - number of time instances.
```

```
23 %   amb_mask - Optional, 2D Array of size [Nx, Ny]. Ambiguity function
24 %               mask, amb = amb * amb_mask.
25 %               (default: amb_mask = ones(Nx, Ny)).
26 %
27 % Outputs:
28 %   tfd - Optional, 2D array [Ntfx_fin, Ntfy_fin]. Time-frequency
29 %         distribution (TFD). If no output is specified function plots
30 %         the resulting TFD.
31 %
32 % ------------------------------------------------------------------------
33 % Copyright (2017): Ivan Volaric
34
35 %------------------------------------------------------------------------
36 % INITALIZIATION
37 %------------------------------------------------------------------------
38     %Default values, if not specified by the user.
39     if (nargin == 0),
40         error('The number of inputs must be at least 1.');
41     elseif (nargin == 1),
42         N = size(amb); Ntf = [N(2)+1-rem(N(1),2),N(1)];
43         Ntf_fin = Ntf; amb_mask = ones(N);
44     elseif (nargin == 2),
45         Ntf = flip(N); Ntf_fin = Ntf; amb_mask = ones(N);
46     elseif (nargin == 3),
47         Ntf_fin = Ntf; amb_mask = ones(N);
48     elseif (nargin==4),
49         amb_mask = ones(N);
50     end;
51
52     %Check if inputs are valid.
53     if (~ismatrix(amb))
54         error('amb must be a matrix.');
55     end
56     [Norg_x, Norg_y] = size(amb);
57     try
58         [N_x,N_y] = ambi2tf_checkinput(N,'N');
59         [Ntf_x,Ntf_y] = ambi2tf_checkinput(Ntf,'Ntf');
60         [Ntf_fin_x,Ntf_fin_y] = ambi2tf_checkinput(Ntf_fin,'Ntf_fin');
61     catch exception
62         throw(exception)
63     end
64
65     if ((N_x > Norg_x) || (N_y > Norg_y))
66         error('N must me smaller or equal than size(amb).');
67     end
68     if ((Ntf_x < N_y) || (Ntf_y < N_x))
69         error('Ntf must me greater or equal than N.');
```

```
70      end
71      if (~ismatrix(amb_mask))
72          error('amb_mask must be a matrix.');
73      end
74      if (~size(amb_mask)==[N_x, N_y])
75          error('amb_mask must be size of [Nx, Ny].');
76      end
77
78  %-------------------------------------------------------------------------
79  % ADJUSMENT OF THE AF DIMENSIONS
80  %-------------------------------------------------------------------------
81     %Reduction of the AF dimensions to the [Nx, Ny] (if larger).
82     if ((N_x < Norg_x) || (N_y < Norg_y))
83          amb_temp = zeros(N_x,N_y);
84          amb_temp(:,:) = amb((1+ceil(Norg_x/2-N_x/2)):1:ceil(Norg_x/2 ...
85          + N_x/2),(1+ceil(Norg_y/2-N_y/2):1:ceil(Norg_y/2+N_y/2)));
86          amb = amb_temp;
87     end
88
89     %Masking of the AF.
90     amb = amb.*amb_mask;
91
92     %AF zero-padding to the [Ntfx, Ntfy] (if smaller).
93      if ((Ntf_x > N_y+rem(N_y,2)) || (Ntf_y > N_x))
94          amb_temp = zeros(Ntf_y,Ntf_x);
95          amb_temp((1+ceil(Ntf_y/2-N_x/2)):1:(ceil(Ntf_y/2+N_x/2)),...
96              (1+ceil(Ntf_x/2-N_y/2)):1:(ceil(Ntf_x/2+N_y/2))) = amb(:,:);
97          amb = amb_temp;
98      end
99
100 %-------------------------------------------------------------------------
101 % DOMAIN TRANSFORMATION
102 %-------------------------------------------------------------------------
103     Ntf_x = max(Ntf_x,2);
104     amb = amb([floor(Ntf_y/2)+1:Ntf_y 1:floor(Ntf_y/2)],:);
105     ambi = ifft(amb).';
106     %Time-delay representation.
107     tdr = zeros(Ntf_x,Ntf_y);
108     tdr((1:ceil(Ntf_x/2)),:) = ambi((ceil(Ntf_x/2)):...
109          (Ntf_x-(~rem(Ntf_x,2))),:);
110     tdr((Ntf_x:-1:floor(Ntf_x/2)+2),:) = ambi((ceil(Ntf_x/2)-1:-1:1),:);
111     %Time-frequency distribution.
112     tfd = real(fft(tdr));
113
114 %-------------------------------------------------------------------------
115 % ADJUSMENT OF THE TFD DIMENSIONS
116 %-------------------------------------------------------------------------
```

```
117     temp_x = max(Ntf_fin_x,Ntf_x); temp_y = max(Ntf_fin_y,Ntf_y);
118     tfd_temp = zeros(temp_x,temp_y);
119     tfd_temp1 = zeros(Ntf_fin_x,Ntf_fin_y);
120     tfd_temp((1+ceil(temp_x/2-Ntf_x/2)):1:ceil(temp_x/2+Ntf_x/2),...
121         (1+ceil(temp_y/2-Ntf_y/2)):1:ceil(temp_y/2+Ntf_y/2)) = tfd(:,:);
122     tfd_temp1(:,:) = tfd_temp((1+ceil(temp_x/2-Ntf_fin_x/2)):1:(ceil...
123         (temp_x/2+Ntf_fin_x/2)),(1+ceil(temp_y/2-Ntf_fin_y/2)):1:ceil...
124         (temp_y/2+Ntf_fin_y/2));
125     tfd = tfd_temp1;
126
127
128     %If no outputs specified -> plot the resulting TFD.
129     if (nargout == 0)
130         ftf = (0.5*(0:Ntf_fin_y-1)/Ntf_fin_y); ttf = 1:Ntf_fin_x;
131         figure; colormap(flipud(gray));
132         imagesc(ttf,ftf,tfd); axis('xy');
133         title('Time-frequency representation');
134         xlabel('Time, t'); ylabel('Normalized Frequency, f');
135         xlim([0 Ntf_fin_x]); ylim([0 0.5]);
136         set(gca,'xtick',floor(Ntf_fin_x*[0 0.5 1]),'ytick',[0 0.25 0.5]);
137     end
138 end
139
140 function [N_x,N_y] = ambi2tf_checkinput(N,Nname)
141     if (isscalar(N))
142         N_x = N; N_y = N;
143     elseif ((isrow(N)) && (length(N)==2))
144         N_x = N(1); N_y = N(2);
145     else
146         error('%c must be a scalar or vector [%c_x, %c_y].', ...
147             Nname, Nname, Nname);
148     end
149     if ((N_x<=0) || (N_y<=0))
150         error('%s must be greater than zero.', Nname);
151     end
152 end
```

```
1  function amb = fun_tf2ambi(tfd, N, Namb, Namb_fin, amb_mask)
2  %function amb = fun_tf2ambi(tfd, N, Namb, Namb_fin, amb_mask)
3  %Transformation from the time-frequency domain to the ambiguity function.
4  %
5  % Inputs:
6  %   tfd      - Required, 2D Array. Time-frequency distribution (TFD).
7  %   N        - Optional, integer or a 2 element array [Nx, Ny]. Number of
8  %              points of the TFD (N <= size(tfd)) to take into the
9  %              account (default: N = size(tfd)).
```

```
10 %               Nx-number of frequency bins. Ny-number of time instances.
11 %   Namb      - Optional, integer or a 2 element array [Nambx, Namby].
12 %               Number of points of the AF (Namb >= N). Before the
13 %               transformation AF is zero-padded (default: Namb = N).
14 %               Nambx-number of doppler bins.
15 %               Namby-number of lag instances.
16 %   Namb_fin - Optional, integer or a 2 element array
17 %               [Nambx_fin, Namby_fin]. The final number of the AF points.
18 %               If Namb_fin > Nam, AF is zero-padded to the Namb_fin. If
19 %               Namb_fin < Namb, AF is cut to the Namb_fin
20 %               (default: Namb_fin = Namb).
21 %               Nambx_fin-number of doppler bins.
22 %               Namby_fin-number of lag instances.
23 %   amb_mask - Optional, 2D Array of size [Nambx_fin, Namby_fin].
24 %               Ambiguity function mask, amb = amb * amb mask.
25 %               (default: amb mask = ones(Nambx_fin, Namby_fin)).
26 %
27 % Outputs:
28 %   amb -       Optional, 2D Array [Nambx_fin, Namby_fin]. Ambiguity
29 %               function (AF). If no output is specified function plots
30 %               the resulting TFD.
31 %
32 % ------------------------------------------------------------------------
33 % Copyright (2017): Ivan Volaric
34
35 %-------------------------------------------------------------------------
36 % INITALIZIATION
37 %-------------------------------------------------------------------------
38     %Default values, if not specified by the user.
39     if (nargin == 0),
40         error('The number of inputs must be at least 1.');
41     elseif (nargin == 1),
42         N = size(tfd); Namb = flip(N); Namb_fin = Namb;
43         amb_mask = ones(Namb_fin);
44     elseif (nargin == 2),
45         Namb = flip(N); Namb_fin = Namb; amb_mask = ones(Namb_fin);
46     elseif (nargin == 3),
47         Namb_fin = Namb; amb_mask = ones(Namb_fin);
48     elseif (nargin==4),
49         amb_mask = ones(Namb_fin);
50     end;
51
52     %Check if inputs are valid.
53     if (~ismatrix(tfd))
54         error('tfd must be a matrix.');
55     end
56     [Norg_x, Norg_y] = size(tfd);
```

```
57      try
58          [N_x,N_y] = tf2ambi_checkinput(N,'N');
59          [Namb_x,Namb_y] = tf2ambi_checkinput(Namb,'Namb');
60          Namb_y=Namb_y-(~rem(Namb_y,2)); %force oddness
61          [Namb_fin_x,Namb_fin_y] = tf2ambi_checkinput(Namb_fin,'Namb_fin');
62          Namb_fin_y=Namb_fin_y-(~rem(Namb_fin_y,2)); %force oddness
63      catch exception
64          throw(exception)
65      end
66
67      if ((N_x > Norg_x) || (N_y > Norg_y))
68          error('N must me smaller or equal than size(tfd).');
69      end
70      if ((Namb_x < N_y) || (Namb_y < N_x - rem(Namb_y,2)))
71          error('Namb must me greater or equal than N.');
72      end
73
74  %-------------------------------------------------------------------------
75  % ADJUSMENT OF THE TFD DIMENSIONS
76  %-------------------------------------------------------------------------
77      %Reduction of the TFD dimensions to the [Nx, Ny] (if larger).
78      if ((N_x < Norg_x) || (N_y < Norg_y))
79          tfd_temp = zeros(N_x, N_y);
80          tfd_temp(:,:) = tfd((1+ceil(Norg_x/2-N_x/2)):1:ceil(Norg_x/2 ...
81          +N_x/2),(1+ceil(Norg_y/2-N_y/2):1:ceil(Norg_y/2+N_y/2)));
82          tfd = tfd_temp;
83      end
84
85      %TFD zero-padding to the [Nambx, Namby] (if smaller).
86      if ((Namb_x > N_y) || (Namb_y > N_x))
87          tfd_temp = zeros(Namb_y, Namb_x);
88          tfd_temp((1+ceil(Namb_y/2-N_x/2)):1:(ceil(Namb_y/2+N_x/2)),(1+...
89          ceil(Namb_x/2-N_y/2)):1:(ceil(Namb_x/2+N_y/2))) = tfd(:,:);
90          tfd = tfd_temp;
91      end
92
93  %-------------------------------------------------------------------------
94  % DOMAIN TRANSFORMATION
95  %-------------------------------------------------------------------------
96      %Doppler-frequency representation.
97      tfdi = ifft(tfd);
98      dfr(1:floor(Namb_y/2),:) = tfdi(floor(length(tfdi(:,1))/2)+2: ...
99          length(tfdi(:,1)),:);
100     dfr(Namb_y:-1:(floor(Namb_y/2)+1),:)= tfdi(ceil(Namb_y/2):-1:1,:);
101     %Ambiguity function.
102     amb = fft(dfr.');
103     amb = amb([ceil(Namb_x/2)+1:Namb_x 1:ceil(Namb_x/2)],:);
```

```
104
105 %-------------------------------------------------------------------------
106 % ADJUSMENT OF THE AF DIMENSIONS
107 %-------------------------------------------------------------------------
108     temp_x = max(Namb_fin_x,Namb_x); temp_y = max(Namb_fin_y,Namb_y);
109     amb_temp = zeros(temp_x,temp_y);
110     amb_temp1 = zeros(Namb_fin_x,Namb_fin_y);
111     amb_temp((1+ceil(temp_x/2-Namb_x/2)):1:ceil(temp_x/2+Namb_x/2),(1+...
112         ceil(temp_y/2-Namb_y/2)):1:ceil(temp_y/2+Namb_y/2)) = amb(:,:);
113     amb_temp1(:,:) = amb_temp((1+ceil(temp_x/2-Namb_fin_x/2)):1: ...
114         (ceil(temp_x/2+Namb_fin_x/2)),(1+ceil(temp_y/2-Namb_fin_y/2)):...
115         1:ceil(temp_y/2+Namb_fin_y/2));
116     amb = amb_temp1;
117
118     %masking of the AF
119     amb = amb .* amb_mask;
120
121
122     %if no outputs specified -> plot the resulting AF
123     if (nargout == 0)
124         tautf = (-Namb_fin_x/2+1):1:(Namb_fin_x/2-1);
125         xitf = (-(Namb_fin_y-rem(Namb_fin_y,2))/2:(Namb_fin_y + ...
126             rem(Namb_fin_y,2))/2-1)/Namb_fin_y;
127         figure; colormap(flipud(gray));
128         imagesc(tautf,xitf,abs(amb)); axis('xy');
129         title('Ambiguity function');
130         xlabel('Lag, \tau'); ylabel('Doppler, \nu');
131         xlim([-Namb_fin_x/2 Namb_fin_x/2]); ylim([-0.5 0 0.5]);
132         set(gca,'xtick',floor(Namb_fin_x/2)*[-1,0,1],...
133             'ytick',[-0.5,-0.25,0,0.25,0.5]);
134     end
135 end
136
137 function [N_x,N_y] = tf2ambi_checkinput(N,Nname)
138     if (isscalar(N))
139         N_x = N; N_y = N;
140     elseif ((isrow(N)) && (length(N)==2))
141         N_x = N(1); N_y = N(2);
142     else
143         error('%c must be a scalar or vector [%c_x, %c_y]', ...
144             Nname, Nname, Nname);
145     end
146     if ((N_x <= 0) || (N_y <= 0))
147         error('%s must be greater than zero', Nname);
148     end
149 end
```

Appendix B

MATLAB Code for Detection of Intersection Between the Cross-terms and the Ambiguity Function Axes

```
function [Ntau, Nnu] = fun_adaptiveCSAF(x, verbose)
%function [Ntau, Nnu] = fun_adaptiveCSAF(x, verbose)
% Detects points on the Ambiguity function (AF) axes where the first pair
% of cross-terms is intersecting the respective axis.
%
% Inputs:
%   x        - Required, 1D array. Signal in the time domain.
%   verbose  - Optional, boolean. If true, function plots the results
%              for debugging (default: plot = false).
%
% Outputs:
%   Ntau - Required, lag instance at which first pair of the cross-terms
%          intersect AF lag axis.
%   Nnu -  Required, doppler bins at which first pair of the cross-terms
%          intersect AF doppler axis.
%
% ------------------------------------------------------------------------
% Copyright (2017): Ivan Volaric

%-------------------------------------------------------------------------
% INITALIZIATION
%-------------------------------------------------------------------------
```

```
23      %Default values, if not specified by the user.
24      if (nargin == 0),
25          error('The number of inputs must be at least 1.');
26      elseif (nargin == 1),
27          verbose = false;
28      end
29
30      %Check if inputs are valid.
31      if (~isvector(x))
32          error('x must be a vector.')
33      end
34      if (verbose == 1)
35          verbose = true;
36      elseif (verbose == 0)
37          verbose = false;
38      end
39      if (~islogical(verbose))
40          error('plot must be a boolean variable.')
41      end
42
43      N = length(x);
44      %Number of allowed transitions from negative to positive values.
45      numt_lim = 1;
46      %Moving average filter options.
47      filt = ones(floor(N/12),1); filt = filt/length(filt);
48      %Final threshold value.
49      tol = 0.2;
50
51  %-------------------------------------------------------------------------
52  % INTERSECTION DETECTION
53  %-------------------------------------------------------------------------
54      %Instantaneous power -> Nnu.
55      %Fourier transform of the instantaneous power.
56      IPft = fftshift(abs(fft(abs(x).^2)));
57      %Normalization + just one half.
58      IPft = IPft/max(IPft); IPft = IPft(floor(N/2+1):N);
59      %Filtering with moving average.
60      IPft_filt = conv(IPft,filt,'same');
61      %Normalized derivative.
62      IPft_filt_diff = diff(IPft_filt,1);
63      IPft_filt_diff = IPft_filt_diff/max(IPft_filt_diff);
64      Nnu = inflectionPoint(IPft_filt_diff, numt_lim, tol);
65
66      %Power spectral density -> Ntau.
67      %Inverse Fourier transform of the power spectral density.
68      PSDft = fftshift(abs(ifft(abs(fft(x)).^2)));
69      %Normalization + just one half.
```

```
70      PSDft = PSDft/max(PSDft); PSDft = PSDft(floor(N/2+1):N);
71      %Filtering with moving average.
72      PSDft_filt = conv(PSDft,filt,'same');
73      %Normalized derivative.
74      PSDft_filt_diff = diff(PSDft_filt,1);
75      PSDft_filt_diff = PSDft_filt_diff/max(PSDft_filt_diff);
76      Ntau = inflectionPoint(PSDft_filt_diff, numt_lim, tol);
77
78      %Readjustment of the bigger parameter.
79      if (Nnu >= Ntau)
80          Nnu = ceil(Nnu*0.9);
81      elseif (Ntau >= Nnu)
82          Ntau = ceil(Ntau*0.9);
83      end
84
85  %-------------------------------------------------------------------------
86  % PLOT THE RESULTS (DEBUG)
87  %-------------------------------------------------------------------------
88      if (verbose)
89          %For comparison with filtered counterparts.
90          IPft_diff = diff(IPft,1);
91          IPft_diff = IPft_diff/max(IPft_diff);
92          PSDft_diff = diff(PSDft,1);
93          PSDft_diff = PSDft_diff/max(PSDft_diff);
94
95          %Figures position and axes.
96          monitor = get(0, 'ScreenSize');
97          tau1 = 0:1:(ceil(N/2)-1); nu1 = (0:1:(ceil(N/2)-1));
98          tau2 = 1:1:(ceil(N/2)-1); nu2 = (1:1:(ceil(N/2)-1));
99
100         %F{PSD/IP} - AF zero lag and doppler slices.
101         figure('Outerposition',[0 monitor(4)-400 400 400]);
102         subplot(2,1,1);
103         plot(tau1,PSDft); grid on; hold on;
104         plot(tau1,PSDft_filt,'r'); legend('Original','Filtered');
105         stem(Ntau,PSDft_filt(Ntau+1),'k.');
106         xlabel('\tau'); xlim([0 N/2-1]); ylabel('F^{-1}\{PSD\}');
107         subplot(2,1,2);
108         plot(nu1,IPft); grid on; hold on;
109         plot(nu1,IPft_filt,'r'); legend('Original','Filtered');
110         stem(Nnu,IPft_filt(Nnu+1),'k.');
111         xlabel('\nu'); xlim([0 N/2-1]); ylabel('F\{IP\}');
112
113         %d F{PSD/IP} - Derivatives of the AF axes.
114         figure('Outerposition',[400 monitor(4)-400 400 400]);
115         subplot(2,1,1);
116         plot(tau2,PSDft_diff); hold on; grid on;
```

```
117         plot(tau2,PSDft_filt_diff, 'r'); legend('Original','Filtered');
118         stem(Ntau+1,PSDft_filt_diff(Ntau+1),'k.');
119         xlabel('\tau'); xlim([0 N/2-1]); ylabel('d F^{-1}\{PSD\}');
120         subplot(2,1,2);
121         plot(nu2,IPft_diff); hold on; grid on;
122         plot(nu2,IPft_filt_diff,'r'); legend('Original','Filtered');
123         stem(Nnu+1,IPft_filt_diff(Nnu+1),'k.');
124         xlabel('\nu'); xlim([0 N/2-1]); ylabel('d F\{IP\}');
125     end
126 end
127
128 function [point] = inflectionPoint(data, numt_lim, tol)
129     N = length(data);
130     numt = 0; %Transition counter
131
132     %Create artificial transition if data starts with zero.
133     if (data(1) < max(data)/100)
134         data(1) = - data(1);
135     end
136
137     %Counting the transitions from negative to positive values.
138     for i = (2:1:N)
139         if ((data(i) > 0) && (data(i-1) <= 0))
140             numt = numt + 1;
141             if (numt >= numt_lim)
142                 break;
143             end
144         end
145     end
146     i=i-1;
147
148     %Local maximum after numt_lim transitions.
149     for j = (i:1:(N-1))
150         if ((data(j) > data(j+1)) && (data(j) > data(j-1)) ...
151                 && (data(j+1) > 0) && (data(j) > tol))
152             point = ceil(0.9*(j-1));
153             break;
154         end
155     end
156
157     %Smallest allowed number of points.
158     point = min(point, round(sqrt(N*2)));
159
160     %Force the number of points to be an odd number.
161     point = point-(~mod(point,2));
162 end
```

Appendix C

MATLAB Code for the FICI Based Sparse Reconstruction Algorithm

```
function [tf_new,time] = fun_bpn_FICI_TwIST(y, A, AT, Zc, Rc, alpha, ...
    beta, eps_th, Nit, regul)
%function [tf_new,time] = fun_bpn_FICI_TwIST(y, A, AT, Zc, Rc, alpha, ...
%    beta, eps_th, Nit, regul)
%Sparse reconstruction algorithm based on the FICI rule.
%
% Inputs:
%   y      - Required, 2D array of observed data.
%   A      - Required, function handle for the function that implements
%            the multiplication by the domain transformation matrix (A*y)
%   AT     - Required, function handle for the function that implements
%            the multiplication by the conjugate of A.
%   Zc     - Required, the ICI threshold value.
%   Rc     - Required, allowed amount of the confidence intervals
%            intersection.
%   alpha  - Optional, relaxation parameter alpha of TwIST
%            (default: alpha = 0.5).
%   beta   - Optional, relaxation parameter beta of TwIST
%            (default: beta = 0.5).
%   eps_th - Optional, stopping threshold. Algorithm stops when relative
%            change in the l2 norm of the estimated solution falls below
%            eps_th (default: eps_th = 1e-3).
%   Nit    - Optional, maximum number of iterations allowed in the
%            main phase of the algorithm (default: Nit = 200).
%   regul  - Optional, regularization function. Can be either 's' for
```

```
26 %            soft-thresholding and l1 norm minimization or 'h' for
27 %            hard-thresholding and l0 norm minimization
28 %            (default: regul = 's').
29 %
30 % Outputs:
31 %   tf_new - Required, solution of the reconstruction algorithm.
32 %   time   - Required, algorithm execution time in ms.
33 %
34 % ------------------------------------------------------------------------
35 % Copyright (2017): Ivan Volaric
36
37 %-------------------------------------------------------------------------
38 % INITALIZIATION
39 %-------------------------------------------------------------------------
40     %Default values, if not specified by the user.
41     if (nargin < 5),
42         error('The number of inputs must be at least 5.');
43     elseif (nargin == 5)
44         alpha = 0.5; beta = 0.5; eps_th = 1e-3;
45         Nit = 200; regul = 's';
46     elseif (nargin == 6)
47         beta = 0.5; eps_th = 1e-3; Nit = 200; regul = 's';
48     elseif (nargin == 7)
49         eps_th = 1e-3; Nit = 200; regul = 's';
50     elseif (nargin == 8)
51         Nit = 200; regul = 's';
52     elseif (nargin == 9)
53         regul = 's';
54     end
55
56     %Check if inputs are valid.
57     if ((~ismatrix(y)) && (~isvector(y)))
58         error('y must be a matrix or a vector.');
59     end
60     if ((~isa(A,'function_handle')) || (~isa(AT,'function_handle')))
61         error('A and AT must be a function handles.');
62     end
63     if (Zc < 0)
64         error('Zc must be a positive number.');
65     end
66     if ((Rc < 0) || (Rc > 1))
67         error('Rc must be between 0 and 1.');
68     end
69     if ((alpha < 0) || (alpha > 1))
70         error('alpha must be between 0 and 1.');
71     end
72     if ((beta < 0) || (beta > 2*alpha))
```

```
        error('beta must be between 0 and 2*alpha.');
    end
    if ((Nit < 0) || (~isinteger(Nit)))
        error('Nit must be a positive integer number.');
    end
    if strcmp(regul, 'soft')
        regul = 's';
    elseif strcmp(regul, 'hard')
        regul = 'h';
    end
    if ((~strcmp(regul, 's')) && (~strcmp(regul, 'h')))
        error('regul must be either "s" or "h".');
    end

%-------------------------------------------------------------------------
% MAIN ALGORITHM LOOP
%-------------------------------------------------------------------------
    tf_last = AT(y); tf_last_last=tf_last; tic;
    for i = 1:1:Nit
        %Update the solution of the previous iteration.
        correct_tf = AT(y - A(tf_last)) + tf_last;
        %Find the threshold value.
        threshold = fun_FICIThreshold(sort(abs(correct_tf...
            (correct_tf~=0)),'ascend'),Zc,Rc);
        %Solution of the current iteration.
        tf_new = (1-alpha)*tf_last_last + (alpha-beta)*tf_last + ...
            beta*wthresh(correct_tf, regul ,threshold);
        %Stoping criterion.
        diff = abs((norm(tf_new,2)^2 - norm(tf_last,2)^2) / ...
            norm(tf_last,2)^2);
        if (diff < eps_th)
            break;
        end
        %Save solutions from the previous two iterations.
        tf_last_last = tf_last; tf_last = tf_new;
    end
    time = toc; % Algorithm execution time.
    tf_new(tf_new<0) = 0; %Set all negatives TFD values to 0.
end
```

```
function [threshold] = fun_FICIThreshold(x, Zc, Rc)
%function [threshold] = fun_FICIThreshold(x, Zc, Rc)
%Internal function of the fun_bpn_FICI_TwIST.m. Returns the threshold
%value based on the fast intersection of the confidence intervals.
%
% Inputs:
```

```
7  %   x    -  Required, 1D array. Elements of the array have to be sorted
8  %           in the ascending order with removed zero entries.
9  %   Zc   -  Required, the ICI threshold value.
10 %   Rc   -  Required, allowed amount of the confidence intervals
11 %           intersection.
12 %
13 % Outputs:
14 %   threshold  -  Required, first element of array x at which amount of
15 %                 the confidence intervals intersection is bellow Rc.
16 %
17 % ------------------------------------------------------------------------
18 % Copyright (2017): Ivan Volaric
19
20 %-------------------------------------------------------------------------
21 % INITALIZIATION
22 %-------------------------------------------------------------------------
23     %Check if inputs are valid.
24     if (nargin < 3),
25         error('The number of inputs must be 3.');
26     end
27     if (~isvector(x))
28         error('x must be a vector.')
29     end
30     if (Zc < 0)
31         error('Zc must be a positive number.');
32     end
33     if ((Rc < 0) || (Rc > 1))
34         error('Rc must be between 0 and 1.');
35     end
36
37     num_s = 0; sum_s = 0;
38     sigma = std(x);
39     Dgmin = NaN; Ddmax = NaN;
40
41 %-------------------------------------------------------------------------
42 % INTERSECTION OF CONFIDENCE INTERVALS
43 %-------------------------------------------------------------------------
44     for i = 1:1:length(x);
45         num_s = num_s + 1;
46         sum_s = sum_s + x(i);
47
48         %Upper and lower boundary of the confidence interval.
49         CI_len = Zc*sigma/sqrt(num_s);
50         Dg = (sum_s/num_s) + CI_len;
51         Dd = (sum_s/num_s) - CI_len;
52
53         %Minimum of upper and maximum of lower boundary.
```

```
54          Dgmin = min(Dgmin, Dg);
55          Ddmax = max(Ddmax, Dd);
56
57          if ((Dgmin-Ddmax)/(2*CI_len) < Rc)
58              threshold = x(i+1);
59              break;
60          end
61      end
62  end
```

Appendix D

MATLAB Code for the Localized Rényi Entropy Based Sparse Reconstruction Algorithm

```
function [tf_new, time] = fun_bpn_LocRenyiReconstruction(y, A, AT, nc, ...
    Km, alpha, beta, eps_th, Nit)
%function [tf_new, time] = fun_bpn_LocRenyiReconstruction(y, A, AT, nc,...
%   Km, alpha, beta, eps_th, Nit)
%Sparse reconstruction algorithm based on the localized Renyi entropy.
%
% Inputs:
%   y      - Required, 2D array of observed data.
%   A      - Required, function handle for the function that implements
%            the multiplication by the domain transformation matrix (A*y)
%   AT     - Required, function handle for the function that implements
%            the multiplication by the conjugate of A.
%   nc     - Required, 1D array of instantaneous number of components.
%   Km     - Optional, relaxation parameter for nc. nc is multiplied
%            with the Km. (default: Km = 7).
%   alpha  - Optional, relaxation parameter alpha of TwIST
%            (default: alpha = 0.5).
%   beta   - Optional, relaxation parameter beta of TwIST
%            (default: beta = 0.5).
%   eps_th - Optional, stopping threshold. Algorithm stops when relative
%            change in the l2 norm of the estimated solution falls below
%            eps_th (default: eps_th = 1e-3).
%   Nit    - Optional, maximum number of iterations allowed in the
%            main phase of the algorithm (default: Nit = 200).
%
```

```
% Outputs:
%   tf_new – Required, solution of the reconstruction algorithm.
%   time   – Required, algorithm execution time in ms.
%
% ------------------------------------------------------------------------
% Copyright (2017): Ivan Volaric

%-------------------------------------------------------------------------
% INITALIZIATION
%-------------------------------------------------------------------------
    %Default values, if not specified by the user.
    if (nargin < 4),
        error('The number of inputs must be at least 4.');
    elseif (nargin == 4)
        Km = 7; alpha = 0.5; beta = 0.5; eps_th = 1e-3; Nit = 200;
    elseif (nargin == 5)
        alpha = 0.5; beta = 0.5; eps_th = 1e-3; Nit = 200;
    elseif (nargin == 6)
        beta = 0.5; eps_th = 1e-3; Nit = 200;
    elseif (nargin == 7)
        eps_th = 1e-3; Nit = 200;
    elseif (nargin == 8)
        Nit = 200;
    end

    %Check if inputs are valid.
    if ((~ismatrix(y)) && (~isvector(y)))
        error('y must be a matrix or a vector.');
    end
    if ((~isa(A,'function_handle')) || (~isa(AT,'function_handle')))
        error('A and AT must be a function handles.');
    end
    if (~isvector(nc))
        error('nc must be a vector.');
    end
    if (Km < 0)
        error('Km must be a positive number.');
    end
    if ((alpha < 0) || (alpha > 1))
        error('alpha must be between 0 and 1.');
    end
    if ((beta < 0) || (beta > 2*alpha))
        error('beta must be between 0 and 2*alpha.');
    end
    if ((Nit < 0) || (~isinteger(Nit)))
        error('Nit must be a positive integer number.');
    end
```

```
73
74
75 %-------------------------------------------------------------------------
76 % MAIN ALGORITHM LOOP
77 %-------------------------------------------------------------------------
78     tf_last = AT(y); tf_last_last = tf_last;
79     nc = ceil(nc*Km) + 1;
80     [~, N1] = size(tf_last);
81
82     tic;
83     for i = 1:1:Nit
84         %Update the solution of the previous iteration.
85         correct_tf = AT(y - A(tf_last)) + tf_last;
86         %Time-slice independent hard thresholding.
87         for j = 1:1:N1
88             [~, tf_ind] = sort(correct_tf(:,j),'descend');
89             correct_tf(tf_ind(nc(j):end),j) = 0;
90         end
91         %Solution of the current iteration.
92         tf_new = (1-alpha)*tf_last_last + (alpha-beta)*tf_last + ...
93             beta*correct_tf;
94         %Stoping criterion.
95         diff = abs((norm(tf_new,2)^2 - norm(tf_last,2)^2) / ...
96             norm(tf_last,2)^2);
97         if (diff < eps_th)
98             break;
99         end
100        %Save solutions from the previous two iterations.
101        tf_last_last = tf_last; tf_last = tf_new;
102    end
103    time = toc;
104 end
```

```
1  function [c_num] = fun_CompNumRenyi(x, x_ref, tfd, alpha, dt)
2  %function [c_num] = fun_CompNumRenyi(x, x_ref, tfd, Renyi_alpha, dt)
3  % Calculates the instantaneous number of components present in the TFD
4  % based on the localized Renyi entropy.
5  %
6  % Inputs:
7  %   x      - Required, 1D array. Signal in time domain.
8  %   x_ref  - Optional, 1D array. The referent signal in time domain.
9  %            (default: x_ref = fmconst(N, 0.1)).
10 %   tfd    - Optional, function handle for calculation of the TFD
11 %            (default: tfd = @(x) fun_emb(x, 0.1 , 0.12)).
12 %   alpha  - Optional, order of the Renyi entropy (default: alpha = 7).
13 %   dt     - Optional, localization parameter defining length of the
```

```
14 %            observed time interval (default: dt = 11).
15 %
16 % Outputs:
17 %   c_num  - Required, instantaneous number of components present
18 %            in the TFD.
19 %
20 % ------------------------------------------------------------------------
21 % Copyright (2017): Ivan Volaric
22
23 %------------------------------------------------------------------------
24 % INITALIZIATION
25 %------------------------------------------------------------------------
26     %Default values, if not specified by the user.
27     N = length(x);
28     if (nargin < 1),
29         error('The number of inputs must be at least 1.');
30     elseif (nargin == 2)
31         x_ref = fmconst(N, 0.1); tfd = @(x) fun_emb(x, 0.1 , 0.12);
32         alpha = 7; dt = 11;
33     elseif (nargin == 3)
34         tfd = @(x) fun_emb(x, 0.1 , 0.12); alpha = 7; dt = 11;
35     elseif (nargin == 4)
36         alpha = 7; dt = 11;
37     elseif (nargin == 5)
38         dt = 11;
39     end
40
41     %Check if inputs are valid.
42     if ((~isvector(x)) || (~isvector(x_ref)))
43         error('x and x_ref must be a vectors.');
44     end
45     if (~isa(tfd,'function_handle'))
46         error('tfd must be a function handle.');
47     end
48     if (alpha < 2)
49         error('alpha must be larger then 2.');
50     end
51     if ((dt < 0) || (~isinteger(dt)))
52         error('dt must be a positive integer number.');
53     end
54
55     nt_val = 0.10; %Noise thresholding parameter.
56     c_num = zeros(N,1);
57     tfd_x = tfd(x); tfd_x_ref = tfd(x_ref);
58
59     %Noise thresholding.
60     tfd_x(tfd_x(:) < nt_val*max(tfd_x(:))) = 0;
```

```
61      tfd_x_ref(tfd_x_ref(:) < nt_val*max(tfd_x_ref(:))) = 0;
62
63      %Minimum energy criterion.
64      min_E_crit = 0.01 * sum(tfd_x(:))/(N/dt);
65
66  %-------------------------------------------------------------------------
67  % MAIN ALGORITHM LOOP
68  %-------------------------------------------------------------------------
69      for p = dt:1:(N-dt+1)
70          if (sum(tfd_x(:,p)) < min_E_crit)
71              %Skip current time instance if minimum energy criterion is
72              %not satisfied.
73              continue;
74          end;
75
76          %Localization of the x around the current time instance.
77          tfd_x_p = zeros(size(tfd_x));
78          tfd_x_p(:,(p-dt+1):1:(p+dt-1)) = tfd_x(:,(p-dt+1):1:(p+dt-1));
79
80          %Localization of the x_ref around the current time instance.
81          tfd_x_ref_p = zeros(size(tfd_x_ref));
82          tfd_x_ref_p(:,(p-dt+1):1:(p+dt-1)) = ...
83              tfd_x_ref(:,(p-dt+1):1:(p+dt-1));
84
85          %Number of components in the current time slice.
86          c_num(p) = abs(2^(fun_TFDPerf_Renyi(tfd_x_p, alpha) - ...
87              fun_TFDPerf_Renyi(tfd_x_ref_p, alpha)));
88      end
89      %Number of components before time instance dt.
90      c_num(1:1:dt) = c_num(dt);
91      %Number of components after time instance N-dt.
92      c_num((N-dt+1):1:N) = c_num(N-dt+1);
93  end
94
95  %Calculates Renyi entropy.
96  function M = fun_TFDPerf_Renyi(x,alpha)
97      x = x/sum(x(:)); %Normalization over energy.
98      M=0;
99      for i = 1:1:numel(x)
100         M = M + (x(i)^(alpha));
101     end
102     M = 1/(1-alpha) * log2(M);
103 end
104
105 %Calculates enhanced modified B distribution.
106 function x_emb = fun_emb(x, alpha, beta)
107     N=length(x);
```

```
108     lag_vector = linspace(-N/2, N/2-1, N-(~mod(N,2)));
109     doppler_vector = linspace(-0.5, 0.5, N);
110     [lag_grid, doppler_grid] = meshgrid(lag_vector, doppler_vector);
111 
112     kernel_emb = abs(gamma(beta+1i*pi*doppler_grid)).^2/(gamma(beta))...
113         .^2.*abs(gamma(alpha+1i*pi*lag_grid/N)).^2/(gamma(beta)).^2;
114 
115     x_amb = ambifunb(x);
116     x_emb = fun_ambi2tf(x_amb.*kernel_emb);
117 end
118 
119 %Generates a sine signal with a constant frequency.
120 function y = fmconst(N, fnorm)
121     t0 = round(N/2); tmt0 = (1:N)-t0;
122     y = exp(1j*2*pi*fnorm*tmt0);
123     y = y/y(t0);
124 end
```

Curriculum vitae

Ivan Volarić was born on December 26, 1987 in Rijeka, Croatia. He began his primary education at the elementary school "Pećine", and completed his secondary education at the high school "Srednja škola za elektrotehniku i računalstvo".

In 2006 he enrolled on the undergraduate study of electrical engineering at the Faculty of Engineering of the University of Rijeka, Croatia. He completed the undergraduate study in 2009 by defending the bachelor's thesis titled: "Digital system for the full-step and half-step control of the unipolar and bipolar step motors" (in Croatian: "Digitalni sustav za koračno i polukoračno upravljanje unipolarnim i bipolarnim koračnim motorima"), receiving the bachelor's (B. Sc.) degree in electrical engineering. In the same year he enrolled on the graduate study of electrical engineering at the same institution. He completed the graduate study in 2011 by defending the master's thesis titled: "Improved LPA-ICI Based Method for Fast Signal Denoising", receiving the master's (M. Sc.) degree in electrical engineering.

Since 2012 he is an employee of the Faculty of Engineering of the University of Rijeka, Croatia as a junior researcher and a teaching assistant. As a teaching assistant, he has been holding exercises from the following courses: Measurements in the Electrical Engineering (2012. - 2014.), Digital Logic (2015. - today), Electronics II (2011. - 2013.), and Signals and Systems (2015. - today) on the Undergraduate University Study of Electrical Engineering; Measurements in the Electrical Engineering ST (2012. - 2014.) on the Undergraduate Vocational Study of Electrical Engineering; Electronics CE (2011. - 2013.), Digital Logic (2015. - today), Signals and Systems (2015. - today), and Computer Aided Measurements (2011. - 2014.) on the Undergraduate University Study of Computer Engineering; Automated Instrumentation (2011. - 2014), and Digital Signal Processing (2015. - today) on the Graduate University Study of Electrical Engineering.

Since 2012 he is a PhD student at the Faculty of Engineering, University of Rijeka. As a junior researcher he participated in the scientific project of the Croatian Ministry of Science, Education and Sports no. 069-0362214-1575 titled "Optimization and Design of Time-Frequency Distributions". His current research interests include time-frequency signal analysis, image and video processing and compressive sensing.

List of publications

Scientific papers in peer-reviewed journals referenced in the CC, SCI, and SCI-Expanded databases:

1. Volarić Ivan; Sučić, Viktor; Srdjan Stanković: "A Data Driven Compressive Sensing Approach for Time-Frequency Signal Enhancement", *Signal Processing*, vol. 141 (2017), pp. 229-239. Available: `https://doi.org/10.1016/j.sigpro.2017.06.013`

2. Volarić, Ivan; Lerga, Jonatan; Sučić, Viktor: "A Fast Signal Denoising Algorithm Based on the LPA-ICI Method for Real-Time Applications",*Circuits, Systems, and Signal Processing*, vol. xx (2017), no. xx, pp. 1 - 17. Available: `http://dx.doi.org/10.1007/s00034-017-0538-1`

3. Volarić, Ivan; Stojković, Nino; Vlahinić, Saša: "Noise and Sensitivity Improvement using SC Filters", *Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije*, vol. 56 (2015), no. 4, pp. 491 - 498. Available: `http://dx.doi.org/10.7305/automatika.2016.01.1035`

Scientific papers in international conferences:

1. Volarić, Ivan; Sučić, Viktor: "On the Noise Impact in the L1 Based Reconstruction of the Sparse Time-Frequency Distributions", *International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*, pp. 1-6, Nov 2016, Graz, Austria.

2. Volarić, Ivan; Sučić, Viktor; Car, Zlatan: "A Compressive Sensing Based Method for Cross-Terms Suppression in the Time-Frequency Plane", *IEEE 15th International Conference on Bioinformatics and Bioengineering (BIBE) 2015*, pp. 1 - 4, Nov 2015, Belgrade, Serbia.

3. Volarić, Ivan; Sučić, Viktor; Jurdana, Irena: "Algorithm Performance Analysis for Reconstruction of Sparse Time-Frequency Distributions from Compressive Sensed Ambiguity Function", *Proceedings of International Conference on Innovative Technologies IN-TECH 2015*, pp. 26 - 29, Sept 2015, Dubrovnik, Croatia.

4. Volarić, Ivan; Stojković, Nino; Vlahinić, Saša: "Noise Improvement using SC Filters", *MEET - Microelectronics, Electronics and Electronic Technology*, pp. 127-132, May 2015, Opatija, Croatia.

5. Volarić, Ivan; Sučić, Viktor: "Signal Sparsity and Compressed Sensing in the Time-Frequency Domain", *Proceedings of International Conference on Innovative Technologies IN-TECH 2014*, pp. 325 - 328, Sept 2014, Leiria, Portugal.

6. Volarić, Ivan; Lerga, Jonatan; Sučić, Viktor; Orović, Irena; Stanković Srdjan: "Modification of the ICI Rule Applied to Signal Denoising", *Proceedings of International Conference on Innovative Technologies IN-TECH 2012*, pp. 97 - 101, Sept 2012, Rijeka, Croatia.