MUSI-6201 — Computational Music Analysis

Part 3.2: Fundamentals II

alexander lerch

Georgia Tech Center for Music Technology
College of Architecture

introduction

overview

- **text book**
  - [*Chapter 2: Fundamentals* (pp. 18–28)]
- **additional reading**
  - Richard G. Lyons, *Understanding Digital Signal Processing*, 3rd, Prentice Hall/Pearson, 2011
- **lecture content**
  - block-based processing
  - correlation
  - Fourier Transform
  - other time-frequency transforms

introduction

overview

- **text book**
  - *Chapter 2: Fundamentals* (pp. 18–28)
- **additional reading**
  - Richard G. Lyons, *Understanding Digital Signal Processing*, 3rd, Prentice Hall/Pearson, 2011

- **lecture content**
  - block-based processing
  - correlation
  - Fourier Transform
  - other time-frequency transforms

introduction

overview

- **text book**
  - *Chapter 2: Fundamentals* (pp. 18–28)
- **additional reading**
  - Richard G. Lyons, *Understanding Digital Signal Processing*, 3rd, Prentice Hall/Pearson, 2011
- **lecture content**
  - block-based processing
  - correlation
  - Fourier Transform
  - other time-frequency transforms

introduction
overview

- **text book**
  - *Chapter 2: Fundamentals* (pp. 18–28)
- **additional reading**
  - Richard G. Lyons, *Understanding Digital Signal Processing*, 3rd, Prentice Hall/Pearson, 2011

- **lecture content**
  - block-based processing
  - correlation
  - Fourier Transform
  - other time-frequency transforms

introduction
overview

- **text book**
  - *Chapter 2: Fundamentals* (pp. 18–28)
- **additional reading**
  - Richard G. Lyons, *Understanding Digital Signal Processing*, 3rd, Prentice Hall/Pearson, 2011
- **lecture content**
  - block-based processing
  - correlation
  - Fourier Transform
  - other time-frequency transforms

audio signals

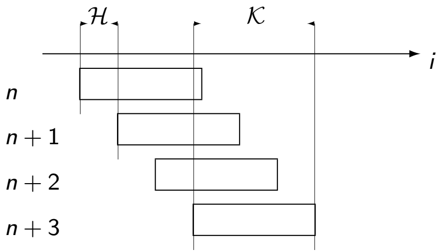
signal categories

block based processing

system implementation

Georgia Tech Center for Music Technology College of Architecture

typical audio applications process chunks of audio data:



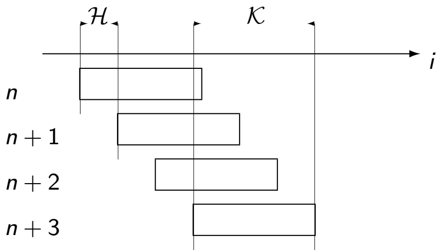
- $\mathcal{K}$: block length
- $\mathcal{H}$: hop length
- n: block index
- i: sample index

block based processing

system implementation

Georgia Tech | Center for Music Technology
College of Architecture

typical audio applications process chunks of audio data:



- $\mathcal{K}$: block length
- $\mathcal{H}$: hop length
- n: block index
- i: sample index

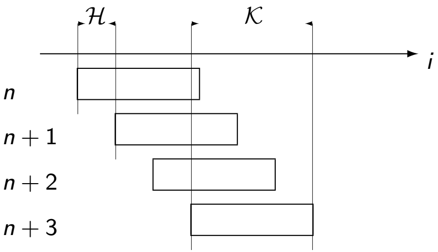
- **reasons**:
  - quasi-stationary signal properties
  - internal block-based processing
  - audio hardware characteristics (real-time systems)
  - efficiency (memory allocation, SIMD)

block based processing

system implementation

Georgia Tech | Center for Music Technology | College of Architecture

typical audio applications process chunks of audio data:



- $\mathcal{K}$: block length
- $\mathcal{H}$: hop length
- n: block index
- i: sample index

- **block boundaries**:

$$\begin{aligned} i_\mathrm{s}(n) &= i_\mathrm{s}(n-1) + \mathcal{H} \\ i_\mathrm{e}(n) &= i_\mathrm{s}(n) + \mathcal{K} - 1 \end{aligned}$$

- **overlap ratio**:

$$o_\mathrm{r} = \frac{\mathcal{K} - \mathcal{H}}{\mathcal{K}}$$

correlation function

introduction

correlation function: compute similarity between two *stationary* signals x,y

$$r_{\mathrm{xy}}(\tau) = \mathcal{E}\{x(t)y(t+\tau)\} \tag{1}$$

- **continuous**:

$$r_{\mathrm{xy}}(\tau) = \int\limits_{-\infty}^{\infty} x(t)\cdot y(t+\tau)dt$$

- **discrete**:

$$r_{\mathrm{xy}}(\eta) = \sum_{i=-\infty}^{\infty} x(i)\cdot y(i+\eta)$$

correlation function

introduction

correlation function: compute similarity between two *stationary* signals x,y

$$r_{\mathrm{xy}}(\tau) = \mathcal{E}\{x(t)y(t+\tau)\} \tag{1}$$

- **continuous**:

$$r_{\mathrm{xy}}(\tau) = \int\limits_{-\infty}^{\infty} x(t)\cdot y(t+\tau)dt$$

- **discrete**:

$$r_{\mathrm{xy}}(\eta) = \sum_{i=-\infty}^{\infty} x(i)\cdot y(i+\eta)$$

correlation function

animation

$$r_{\mathrm{xy}}(\tau) = \int\limits_{-\infty}^{\infty} x(t) \cdot y(t+\tau) dt$$

$$r_{\mathrm{xy}}(\eta) = \sum_{i=-\infty}^{\infty} x(i) \cdot y(i+\eta)$$

matlab source: matlab/animateCorrelation.m

correlation function

examples

draw the correlation function for

matlab source: matlab/displayCorrelation.m

correlation function

examples

draw the correlation function for

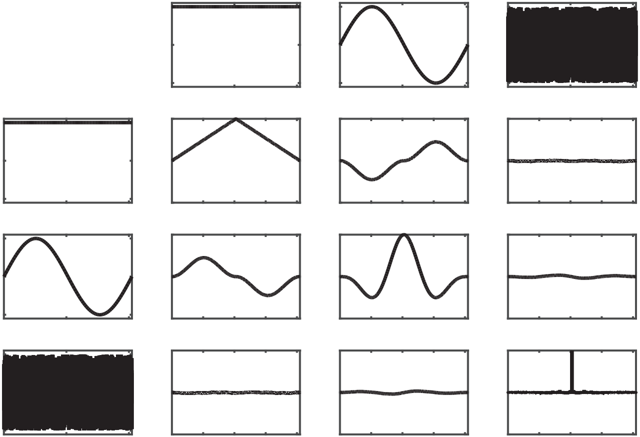
- rectangular window vs.
- sine vs.
- noise

matlab source: matlab/displayCorrelation.m

correlation function

examples

draw the correlation function for



matlab source: matlab/displayCorrelation.m

correlation function

blocked correlation: animation

matlab source: matlab/animateBlockedCorrelation.m

correlation function

normalization

$$\lambda_c = \frac{1}{\sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}$$

avoiding the triangular shape for blocked correlation:

1. modified normalization

$$\lambda_c(\eta) = \frac{\mathcal{K}}{(\mathcal{K} - |\eta|) \cdot \sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}.$$

2. different block lengths $(\mathcal{K}, 3\mathcal{K})$
3. circular application

correlation function

normalization

$$\lambda_c = \frac{1}{\sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right)\cdot\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}$$

avoiding the triangular shape for blocked correlation:

1. modified normalization

$$\lambda_c(\eta) = \frac{\mathcal{K}}{(\mathcal{K}-|\eta|)\cdot\sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right)\cdot\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}.$$

2. different block lengths $(\mathcal{K}, 3\mathcal{K})$
3. circular application

correlation function

normalization

$$\lambda_c = \frac{1}{\sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}$$

avoiding the triangular shape for blocked correlation:

1. modified normalization

$$\lambda_c(\eta) = \frac{\mathcal{K}}{(\mathcal{K} - |\eta|) \cdot \sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}.$$

2. different block lengths $(\mathcal{K}, 3\mathcal{K})$
3. circular application

correlation function

normalization

$$\lambda_c = \frac{1}{\sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}$$

avoiding the triangular shape for blocked correlation:

1. modified normalization

$$\lambda_c(\eta) = \frac{\mathcal{K}}{(\mathcal{K} - |\eta|) \cdot \sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}.$$

2. different block lengths $(\mathcal{K}, 3\mathcal{K})$
3. circular application

correlation function

normalization

$$\lambda_c = \frac{1}{\sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}$$

avoiding the triangular shape for blocked correlation:

1. modified normalization

$$\lambda_c(\eta) = \frac{\mathcal{K}}{(\mathcal{K} - |\eta|) \cdot \sqrt{\left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} x^2(i)\right) \cdot \left(\sum\limits_{i=i_\mathrm{s}(n)}^{i_\mathrm{e}(n)} y^2(i)\right)}}.$$

2. different block lengths $(\mathcal{K}, 3\mathcal{K})$
3. circular application

autocorrelation function

definition & properties

correlation function with the signal itself

!

autocorrelation function properties

- ACF at lag 0:
  $r_{xx}(0, n) = 1$ if normalized, RMS otherwise
- maximum:
  $|r_{xx}(\eta, n)| \leq r_{xx}(0, n)$
- symmetry:
  $r_{xx}(\eta, n) = r_{xx}(-\eta, n)$
- periodicity:
  The ACF of a periodic signal is periodic (period length of input signal)

matlab exercise: correlation

1. implement a Matlab function that computes the ACF for an arbitrary block length
2. consider only one half of the ACF and detect that highest local max that is not the absolute max
3. compute this ACF with overlapping blocks for the audio file *sax_example.wav*
4. plot that lag of the detected maxima over blocks and discuss the results

summary

lecture content