Digital Signal Processing for Music

Part 23: Time-stretching and Pitch-shifting

Andrew Beck

Intro

- **Time Stretching**

  Change playback speed/tempo without changing pitch

- **Pitch Shifting**

  Change pitch without changing tempo / playback speed

- **Terms**
  - Time / pitch scaling
  - Time expansion / compression

Applications

- **Beat Matching:**: Align tempo of two or more audio files (mashup)
- **Key lock**: "Align" pitch of two or more audio files (mashup)
- **Pitch / time correction**: Edit intonation, frequency deviation, vibrato, glissando
- Video **frame rate conversion**
- **Sample player / libraries**
- **Sound design**
- **Educational software**: Pitch and timing visualization

Stretch and Pitch Factors

$$s = \frac{t_{output}}{t_{input}} \qquad p = \frac{f_{output}}{f_{input}}$$

Examples:

- *Half speed:* $s = 2$
- *Half pitch:* $p = \frac{1}{2}$
- *Semitone up / down:*

$$p_u = 2^{\frac{1}{12}} = 1.059 \quad p_d = 2^{-\frac{1}{12}} = 0.9439$$

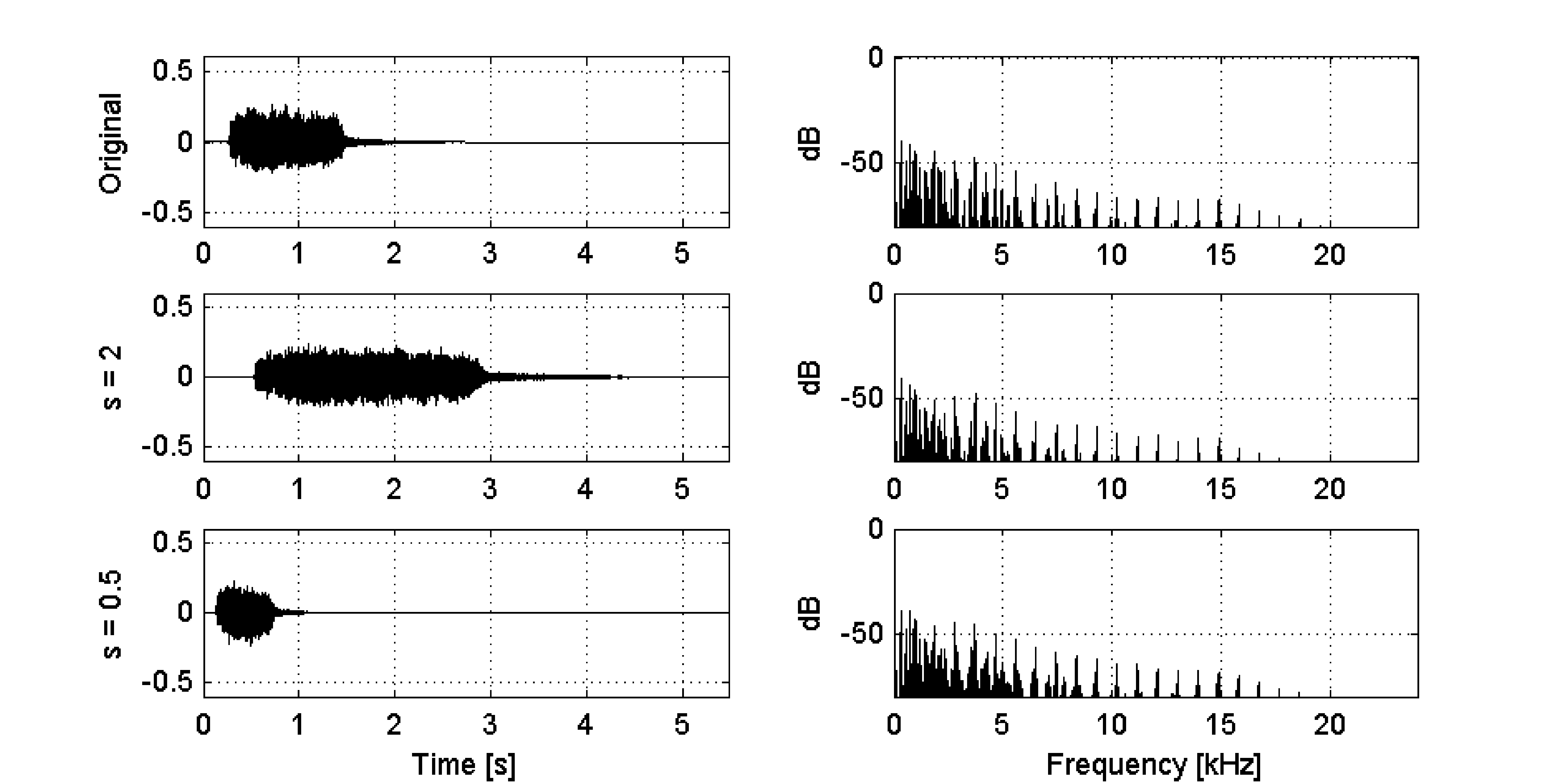
- *100 BPM → 75 BPM:* $s = \frac{4}{3}$

Resampling

- **Traditional**: resampling
  - Change inter-sample 'distance' by interpolation
  - Keep playback sample rate constant
  - Audio example
    - Original: 0:00 / 0:15
    - Resample: 0:00 / 0:11
- Tempo changes results in pitch change (and vice versa)

$$s = \frac{1}{p}$$

Stretching: Effect on Frequency Domain



OLA: Introduction

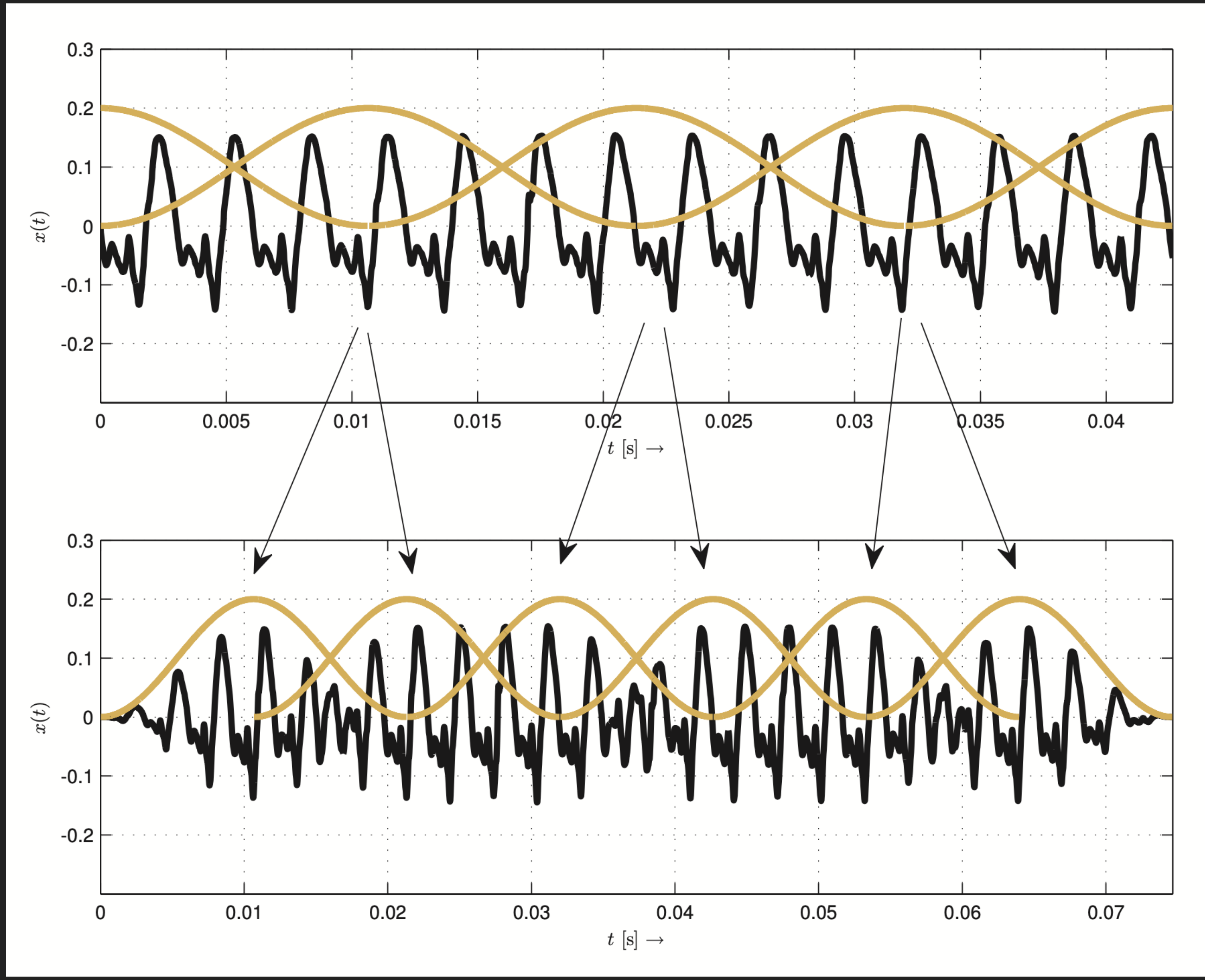
Overlap and add approaches for:

- Granular synthesis
- Time /frequency synthesis and processing
- Time-stretching and pitch-shifting

Time Stretching

Overlap and add

1. **Split input** signal into overlapping blocks
2. **Duplicate or discard blocks** depending on stretch factor



» Original:

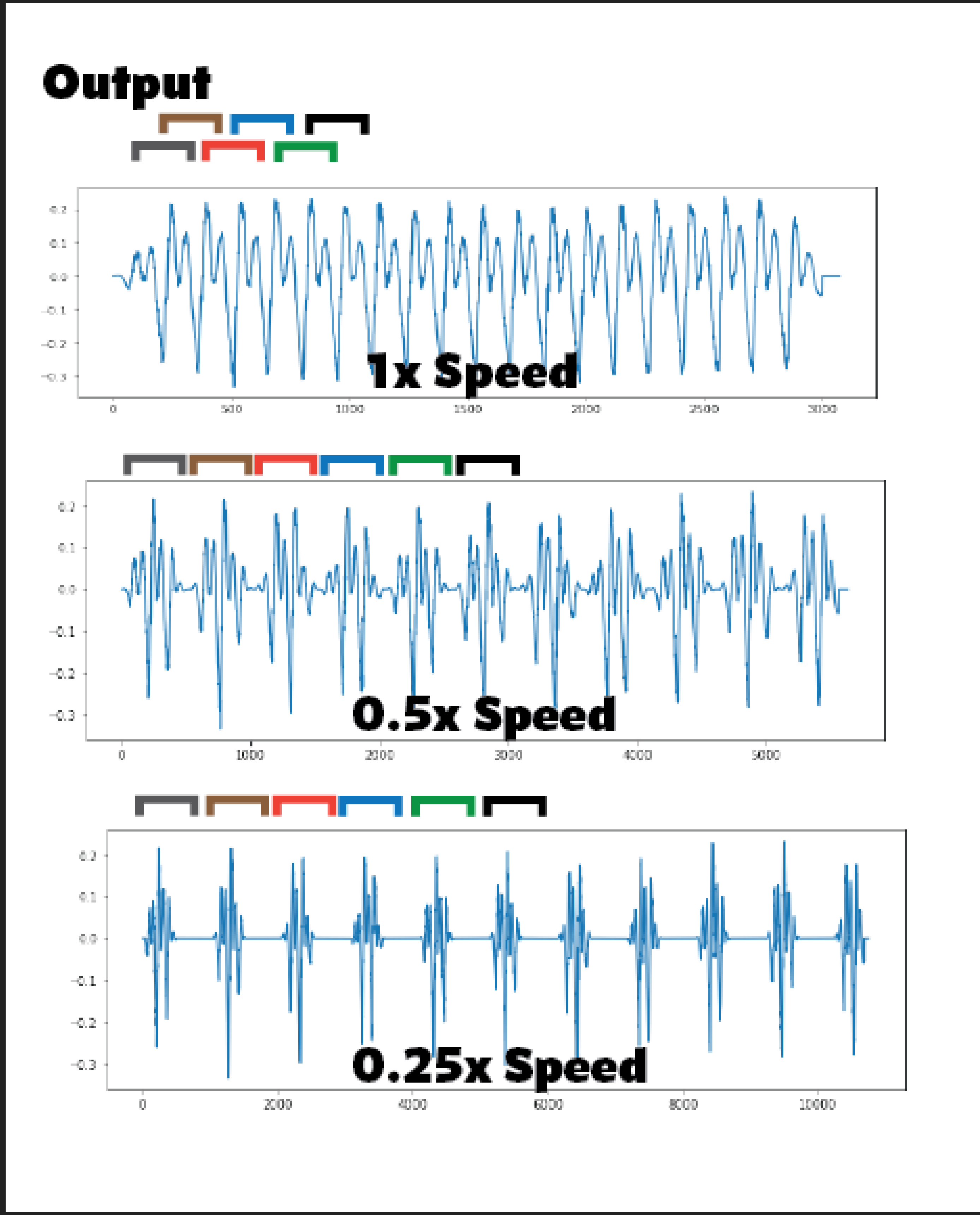
0:00 / 0:15

» $s = \frac{4}{3}$:

0:00 / 0:20

Windowed Grains

Outputing Grains

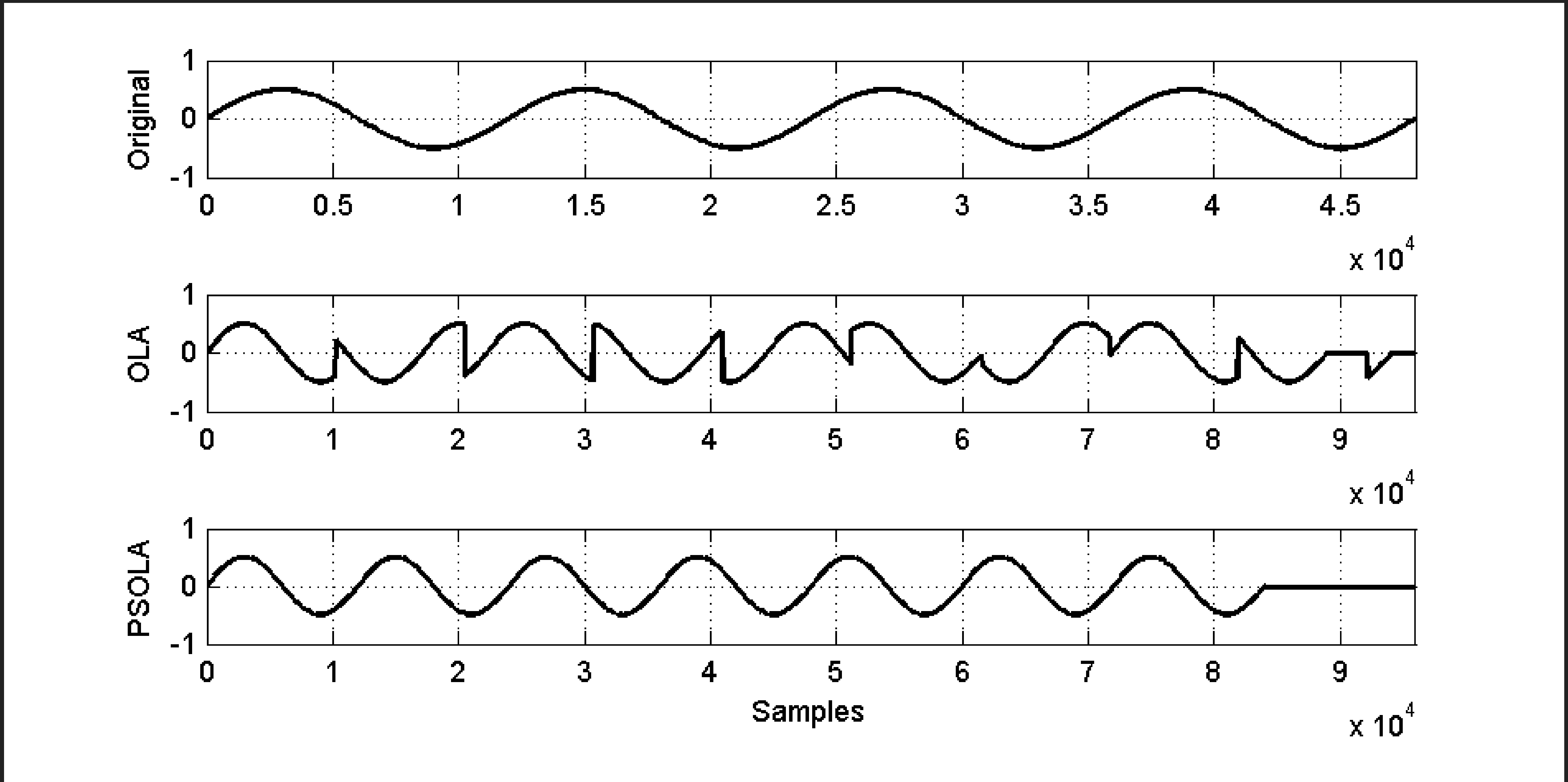


Parameters

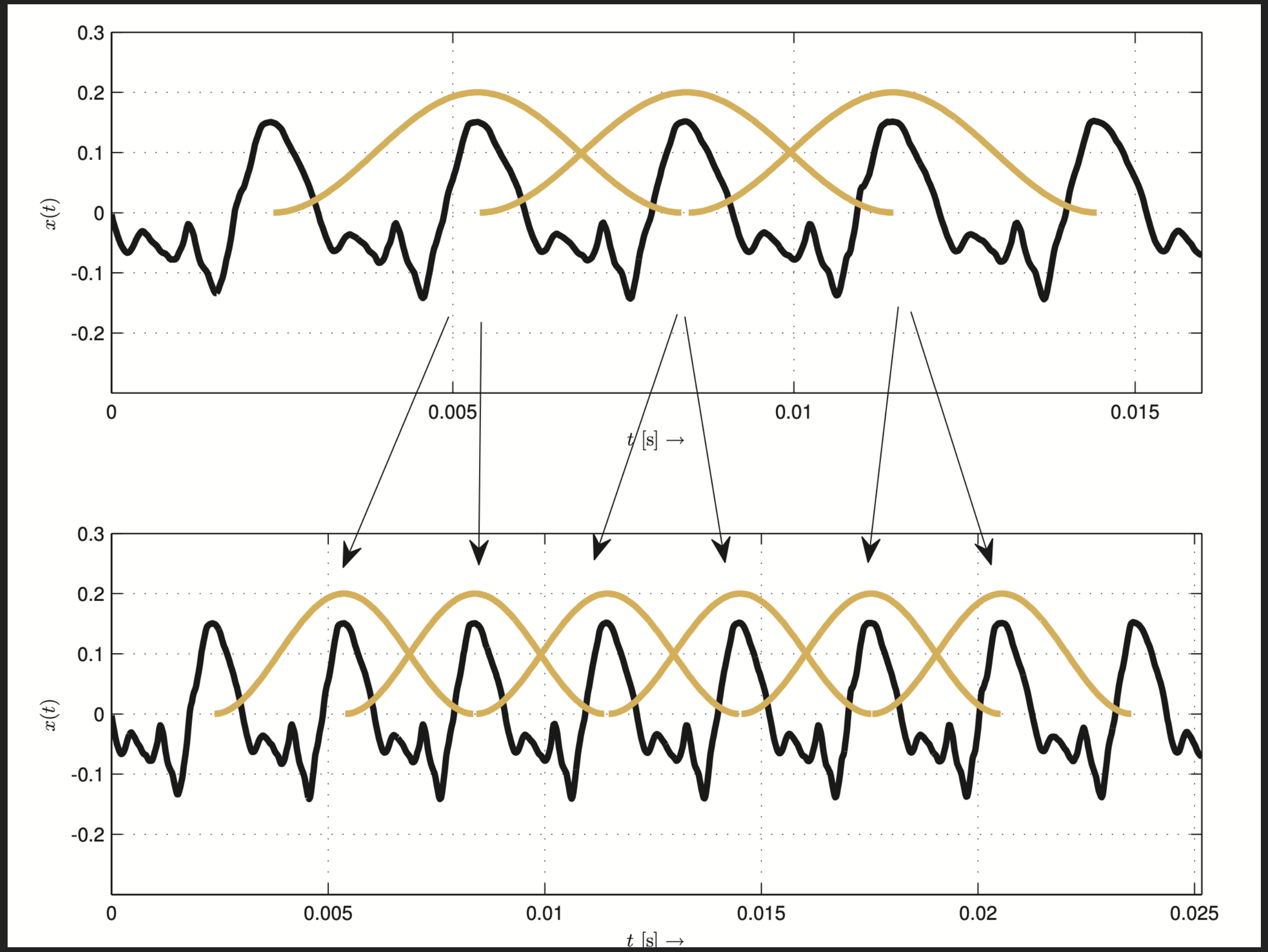
- Grain Size
- Hop Size
- Time Scale
- Pitch Scale
- Time Variance
- Pitch Variance
- Interpolation Method

Pitch Synchronous Overlap and Add

- Use the OLA principle, but
- **Adapt block length** to fundamental period length



PSOLA: Example



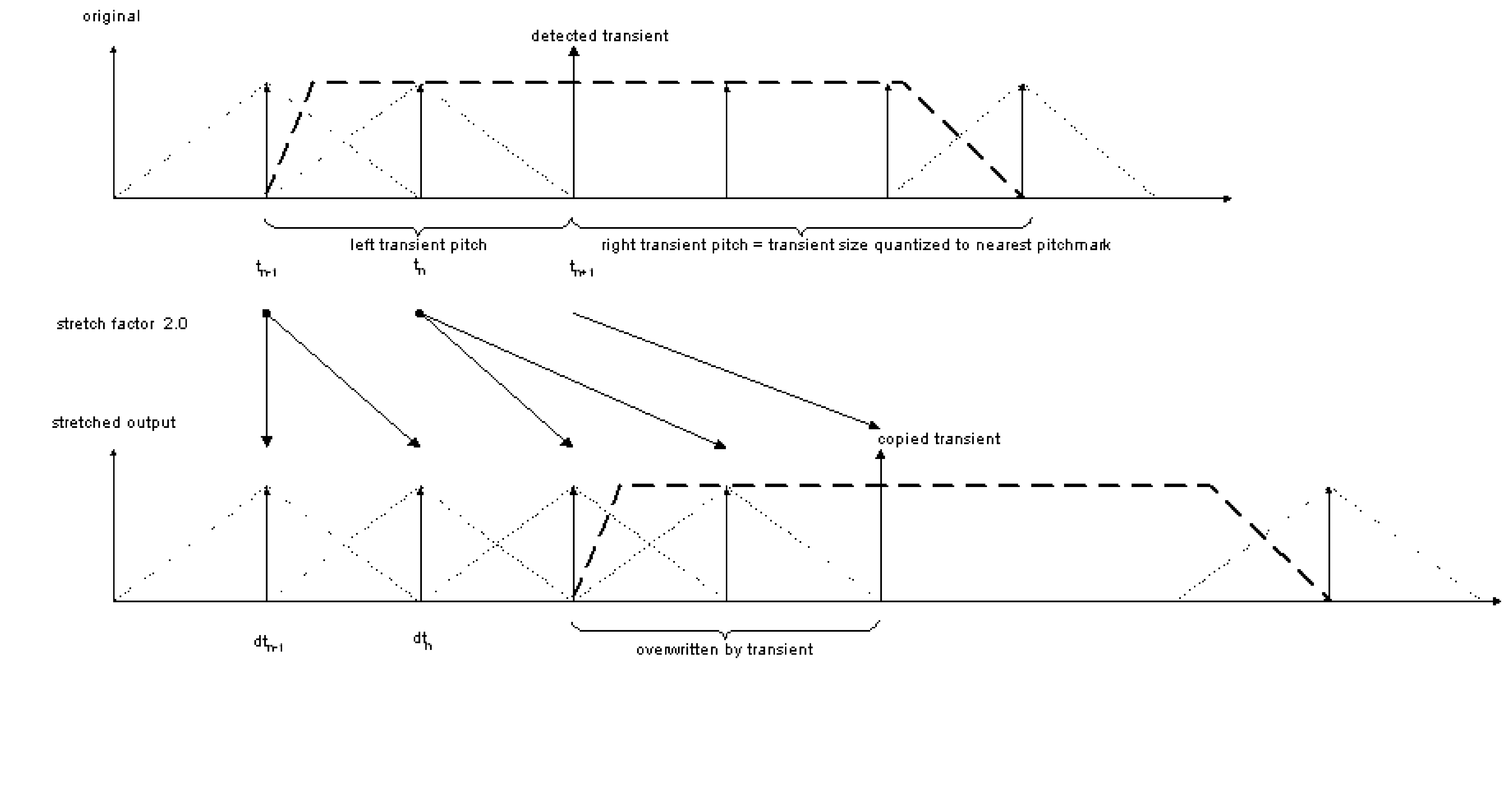
» Original:

0:00 / 0:15

» $s = \frac{4}{3}$:

0:00 / 0:20

PSOLA: Transient Copying



PSOLA: Summary

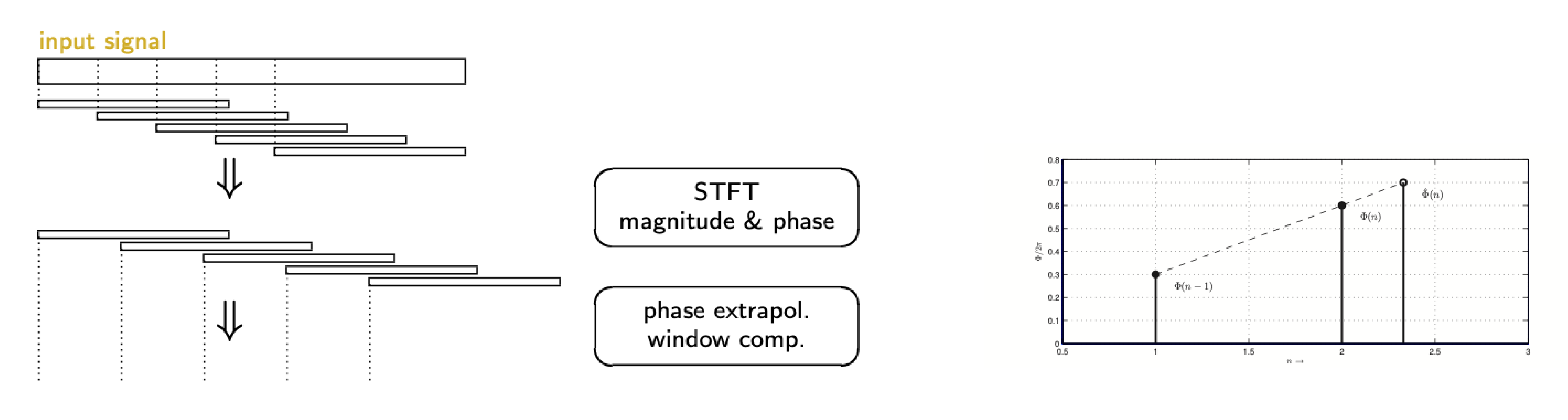
- **Processing Steps**
  Detect *fundamental frequency* / period length
  Set *pitch marks*
  Intelligently *select blocks* to be repeated / discarded
- **Advantages**
  - *High granularity* - Modify audio on period length resolution
  - *High quality*

PSOLA: Summary

- **Problems**
  - Quality depends on *pitch tracking* reliability
  - Quality and timbre depends on *pitch mark positioning*
  - Works only for *monophonic* input signals
    - Polyphonic and noisy segments
    - Reverberation and overlapping tones
  - *Noise, plosives* require special consideration
  - *Copying* artifacts (double transients, timing deviations)
- **Typical Applications**
  - **Standard approach for vocal editing tools**

Phase Vocoder: Frequency Domain OLA

1. **Split input** signal into overlapping blocks
2. Compute **magnitude and phase spectrum** of each block
3. **Change overlap ratio** between blocks depending on stretch factor
4. Keep the magnitude, **adapt the phase per bin** to the blocks new time stamp



Phase Vocoder: Audio Example

- Original: 0:00 / 0:15
- Resample: 0:00 / 0:20

Frequency Reassignment: Relation of Phase and Frequency

Phasor Representation:

- Sine value is defined by magnitude and phase
- Decreasing the amplitude → shorter vector
- Increasing the frequency → Increasing speed

Frequency and phase change closely related

- Time for full rotation is period length T with

$$f = \frac{1}{T}$$

- Time for fractional rotation $\Delta\Phi$ is corresponding fraction of period length

$$f = \frac{\Delta\Phi}{\Delta t}$$

Frequency and phase change closely related

- In other words:

$$\Phi(t) = \omega \cdot t$$

$$\Rightarrow \frac{d\Phi(t)}{dt} = \omega = 2\pi f$$

Frequency Reassignment: Principles

Frequency domain

- Instead of using the bin frequency

$$f(k) = k * \frac{f_\mathrm{S}}{\mathcal{K}}$$

- We use the phase of each bin $\Phi(k, n)$
- To compute the frequency from the phase difference of neighboring blocks

$$\omega_\mathrm{I}(k, n) \propto \Phi(k, n) - \Phi(k, n-1)$$

- $\omega_\mathrm{I}(k, n)$ is called **instantaneous frequency** per block per bin

Frequency Reassignment: Scaling Factor

- Instantaneous frequency calculation has to take into account
  - Hop Size: $\mathcal{H}$
  - Sample Rate: f_s

$$\omega_{\mathrm{I}}(k,n) = \frac{\Delta\Phi_{\mathrm{u}}(k,n)}{\mathcal{H}} \cdot f_{\mathrm{S}}$$

- Problem: Phase ambiguity

$$\Phi(k,n) = \Phi(k,n) + j \cdot 2\pi$$

- *Phase unwrapping*

Phase Unwrapping

- Compute unwrapped phase $\Phi_{\mathrm{u}}(k,n)$
  - Estimate unwrapped bin phase

$$\hat{\Phi}(k,n) = \Phi(k,n-1) + \underbrace{2\pi k \cdot \frac{\mathcal{H}}{\mathcal{K}}}_{=\omega_k \cdot \frac{\mathcal{H}}{f_{\mathrm{s}}}}$$

  - Unwrap phase by shifting current phase to estimates range

$$\Phi_{\mathrm{u}}(k,n) = \hat{\Phi}(k,n) + \operatorname{princarg}\Big[\Phi(k,n) - \hat{\Phi}(k,n)\Big]$$

Phase Unwrapping

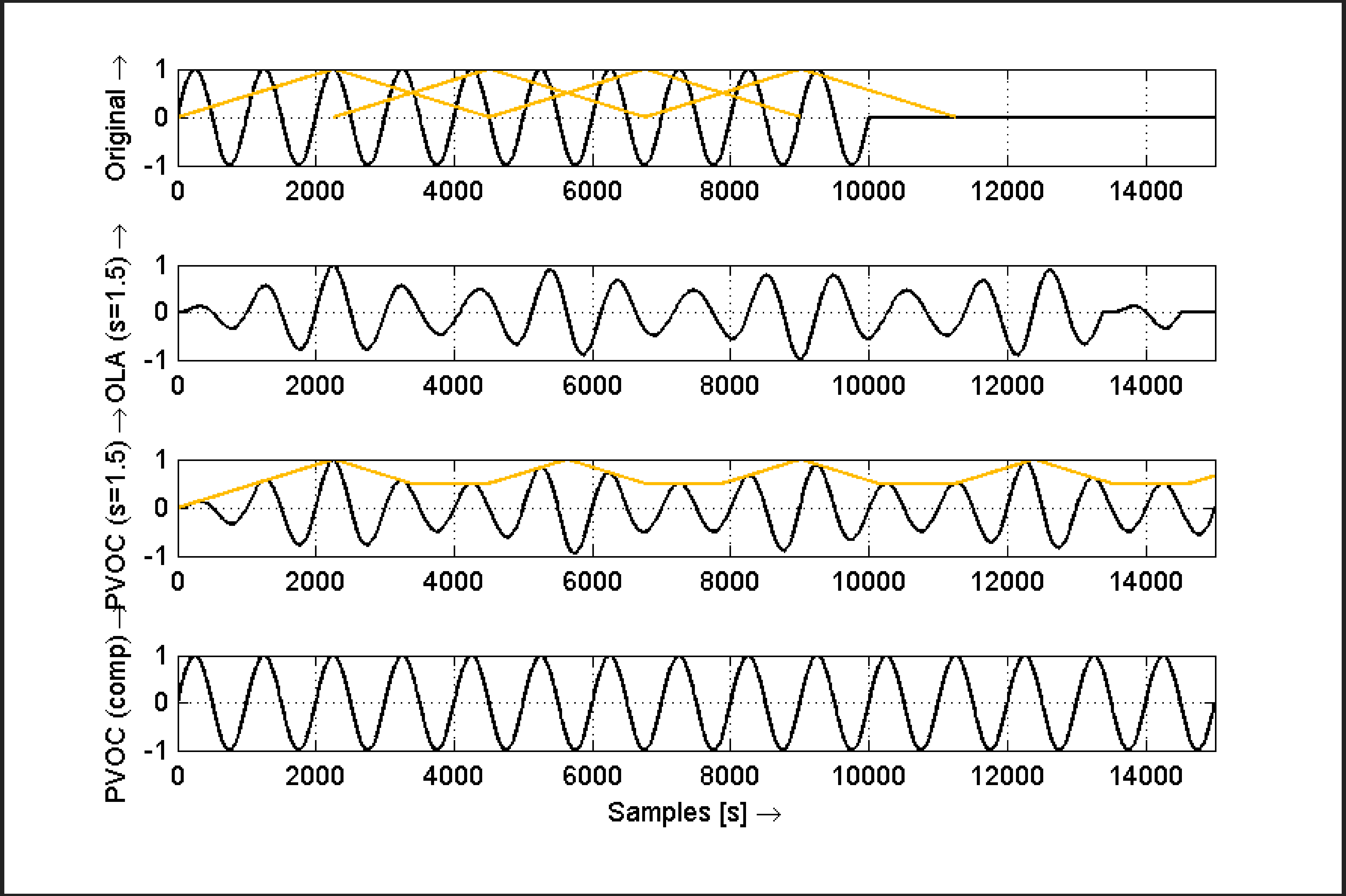
- Compute unwrapped phase difference

$$
\begin{aligned}
\Delta\Phi_{\mathrm{u}}(k,n) &= \Phi_{\mathrm{u}}(k,n) - \Phi(k,n-1) \\
&= \hat{\Phi}(k,n) + \operatorname{princarg}\left[\Phi(k,n) - \hat{\Phi}(k,n)\right] - \Phi(k,n \\
&= \frac{2\pi k}{\mathcal{K}}\mathcal{H} + \operatorname{princarg}\Bigg[\Phi(k,n) - \Phi(k,n-1) - \frac{2\pi}{\mathcal{K}}
\end{aligned}
$$

Frequency Reassignment: Problems

- **Overlapping spectral components**
  - Sinusoidal components often overlap (Spectral leakage, several instruments playing the same pitch, ...)
  - Incorrect phase estimates
  - Spectrum should be as sparse as possible, increase STFT length
- **Inaccurate phase unwrapping**

Phase Vocoder Window Compensation



Phase Vocoder - Properties & Artifacts

- **Advantages**
  - Allows *polyphonic input* (assumption: no overlapping harmonics)
  - Absolute *timing stability* (i.e., sample resolution)
- **Disadvantages**
  - *Low granularity* - FFT block size
  - Artifacts: Phasing, Transient smearing / doubling