pvx Complete HTML Documentation

Total HTML pages included: **23**

References appendix: **143** papers (minimum enforced: 100).

Generated: 2026-02-20 18:28:28 UTC.

Generated by `scripts_generate_docs_pdf.py`.

Contents

pvx HyperText Markup Language (HTML) Documentation

Source: index.html

This HTML documentation is organized by algorithm folder/theme and generated from `src/pvx/algorithms/registry.py` and `src/pvx/algorithms/base.py` dispatch parameters.

Totals:

- **111** algorithms
- **12** folders/themes
- **87** distinct algorithm parameter keys in dispatch
- **103** linked technical glossary concepts
- **143** bibliography references

Quick Setup (Install + PATH)

```
python3 -m venv .venv
source .venv/bin/activate
python3 -m pip install -e .
pvx --help
```

If `pvx` is not found, add the virtualenv binaries to your shell path (`zsh`):

```
printf 'export PATH="%s/.venv/bin:$PATH"\n' "$(pwd)" >> ~/.zshrc
source ~/.zshrc
pvx --help
```

No-PATH fallback: `python3 pvx.py voc input.wav --stretch 1.2 --output output.wav`

Research bibliography (143 references) - foundational and directly related literature for phase vocoder, time-scale/pitch, time-frequency transforms, separation, denoising, dereverberation, spatial audio, and mastering workflows.

Linked technical glossary (103 terms) - quick definitions and external references (Wikipedia, standards pages, docs, and papers).

Mathematical foundations and **window reference (all 50 windows)**.

Architecture diagrams, **algorithm limitations**, **benchmark report**, **pipeline cookbook**, **CLI flag index**, and **citation quality report**.

GitHub usually shows `.html` files as source. For browser-rendered math on GitHub itself, use the Markdown mirrors: `docs/MATHEMATICAL_FOUNDATIONS.md` and `docs/WINDOW_REFERENCE.md`.

Algorithm Groups

Folder	Theme	Algorithms	Page
`time_scale_and_pitch_core`	Time-Scale and Pitch Core	7	Open
`pitch_detection_and_tracking`	Pitch Detection and Tracking	8	Open
`retune_and_intonation`	Retune and Intonation	8	Open
`spectral_time_frequency_transforms`	Spectral and Time-Frequency Transforms	8	Open
`separation_and_decomposition`	Separation and Decomposition	8	Open
`denoise_and_restoration`	Denoise and Restoration	8	Open
`dereverb_and_room_correction`	Dereverb and Room Correction	8	Open
`dynamics_and_loudness`	Dynamics and Loudness	8	Open
`creative_spectral_effects`	Creative Spectral Effects	8	Open
`granular_and_modulation`	Granular and Modulation	8	Open

Folder	Theme	Algorithms	Page
`analysis_qa_and_automation`	Analysis, QA, and Automation	8	Open
`spatial_and_multichannel`	Spatial and Multichannel	24	Open

pvx Architecture

Source: architecture.html

Architecture overview for runtime processing, algorithm dispatch, documentation generation, and CI/Pages publication.

Runtime and CLI Flow

Algorithm Registry and Dispatch

```
flowchart TD
  R[src/pvx/algorithms/registry.py] --> B[src/pvx/algorithms/base.py]
  B --> M1[time_scale_and_pitch_core/*]
  B --> M2[retune_and_intonation/*]
  B --> M3[dynamics_and_loudness/*]
  B --> M4[spatial_and_multichannel/*]
```

Documentation Build Graph

```
flowchart LR
  G1[scripts_generate_python_docs.py] --> D[docs/*]
  G2[scripts_generate_theory_docs.py] --> D
  G3[scripts_generate_docs_extras.py] --> D
  G4[scripts_generate_html_docs.py] --> H[docs/html/*]
  D --> H
```

CI and Pages

```
flowchart LR
  PR[Push / PR] --> CI[Doc and test workflow]
  CI --> S[Generation + drift checks]
  S --> T[Unit tests + docs coverage tests]
  T --> P[GitHub Pages deploy workflow]
  P --> SITE[Published docs site]
```

pvx Mathematical Foundations

Source: math.html

Mathematical summary of the core signal-processing model used across pvx. Equations are rendered with MathJax when this page is opened in a normal browser.

GitHub usually displays HTML source text. For GitHub-native math rendering, use `docs/MATHEMATICAL_FOUNDATIONS.md`.

STFT Analysis and Synthesis

Analysis:

$$X_t[k] = \sum_{n=0}^{N-1} x[n + tH_a] w[n] e^{-j2\pi kn/N}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Each frame t is windowed by $w[n]$ and transformed to complex bins k.

Synthesis:

$$\hat{x}[n] = \frac{\sum_t \hat{x}_t[n - tH_s] w[n - tH_s]}{\sum_t w^2[n - tH_s] + \varepsilon}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Overlap-add with energy normalization preserves level stability.

Transform Backend Selection (`--transform`)

pvx lets you choose the per-frame transform backend used in analysis/resynthesis: `fft`, `dft`, `czt`, `dct`, `dst`, and `hartley`.

Fourier family (`fft`, `dft`):

$$X_t[k] = \sum_{n=0}^{N-1} x_t[n] e^{-j2\pi kn/N}, \qquad x_t[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_t[k] e^{j2\pi kn/N}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Chirp-Z (`czt`):

$$X_t[k] = \sum_{n=0}^{N-1} x_t[n] A^{-n} W^{nk}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

With pvx defaults $A = 1$ and $W = e^{-j2\pi/N}$, CZT samples the DFT contour via a different numerical path.

DCT-II (`dct`):

$$C_t[k] = \alpha_k \sum_{n=0}^{N-1} x_t[n] \cos\left(\frac{\pi}{N}(n + \tfrac{1}{2})k\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

DST-II (`dst`):

$$S_t[k] = \beta_k \sum_{n=0}^{N-1} x_t[n] \sin\left(\frac{\pi}{N}(n+\tfrac{1}{2})(k+1)\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Hartley (`hartley`):

$$H_t[k] = \sum_{n=0}^{N-1} x_t[n] \operatorname{cas}\left(\frac{2\pi kn}{N}\right), \ \operatorname{cas}(\theta) = \cos\theta + \sin\theta$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Transform	Why use it	Tradeoffs	Example use case
`fft`	Fastest and most robust default; best runtime and CUDA support.	Typical STFT leakage/time-resolution tradeoffs still apply.	General production stretch/pitch workflows.
`dft`	Reference Fourier baseline for transform parity checks.	Usually slower than `fft` with little audible benefit.	Verification and algorithm A/B testing.
`czt`	Alternative numerical path for awkward or prime frame sizes.	Requires SciPy; generally slower and CPU-oriented.	Edge-case frame-size validation and diagnostics.
`dct`	Real-basis energy compaction, useful for envelope-focused shaping.	No explicit complex phase bins; less transparent for strict phase coherence.	Creative timbre shaping and coefficient-domain experiments.
`dst`	Odd-symmetry real-basis alternative with distinct coloration.	Same phase limitations as DCT; artifacts are content-dependent.	Experimental percussive/spectral texture variants.
`hartley`	Real transform with Fourier-related basis for exploratory comparisons.	Different bin semantics from complex STFT can change artifact character.	Pedagogical and creative real-domain phase-vocoder tests.

Sample use cases

```
python3 pvxvoc.py dialog.wav --transform fft --time-stretch 1.08 --transient-preserve --output-dir out
python3 pvxvoc.py tone_sweep.wav --transform dft --time-stretch 1.00 --output-dir out
python3 pvxvoc.py archival_take.wav --transform czt --n-fft 1531 --win-length 1531 --hop-size 382 --output-dir out
python3 pvxvoc.py strings.wav --transform dct --pitch-shift-cents -17 --output-dir out
python3 pvxvoc.py percussion.wav --transform dst --time-stretch 0.92 --phase-locking off --output-dir out
python3 pvxvoc.py synth_pad.wav --transform hartley --time-stretch 1.30 --output-dir out
```

Phase-Vocoder Propagation

$$\Delta\phi_t[k] = \operatorname{princarg}\Big(\phi_t[k] - \phi_{t-1}[k] - \omega_k H_a\Big)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

$$\hat{\omega}_t[k] = \omega_k + \frac{\Delta\phi_t[k]}{H_a}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

$$\hat{\phi}_t[k] = \hat{\phi}_{t-1}[k] + \hat{\omega}_t[k] H_s$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

pvx uses these relationships to estimate true instantaneous frequency and re-accumulate phase under a new synthesis hop.

Pitch and Microtonal Mapping

$$r_{\text{pitch}} = 2^{\Delta s/12} = 2^{\Delta c/1200}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Semitone shift Δs and cents shift Δc are equivalent ratio controls. Scale-constrained retuning maps detected F0 to the nearest permitted scale target.

Dynamics and Loudness

$$g_{\text{LUFS}} = 10^{(L_{\text{target}} - L_{\text{in}})/20}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

$$y[n] = x[n] \cdot g_{\text{LUFS}}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Target-loudness gain is applied in linear amplitude domain after dynamics stages, before limiting/clipping safety stages.

Spatial and Multichannel Highlights

Inter-channel phase difference:

$$\Delta\phi_{ij}(k,t) = \phi_i(k,t) - \phi_j(k,t)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Coherence-preserving objective:

$$J = \sum_{k,t} \left| \Delta\phi_{ij}^{out}(k,t) - \Delta\phi_{ij}^{in}(k,t) \right|$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Delay estimate:

$$\tau^* = \arg\max_{\tau} \mathcal{F}^{-1} \left\{ \frac{X_i(\omega) X_j^*(\omega)}{|X_i(\omega) X_j^*(\omega)| + \varepsilon} \right\} (\tau)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

pvx spatial modules focus on channel coherence, stable image cues, and phase-vocoder-consistent multichannel processing for robust chain composition.

pvx Window Reference

Source: windows.html

Complete pvx analysis-window reference. This page covers all **50** supported windows from `pvx.core.voc.WINDOW_CHOICES`, with formula-family mapping and practical interpretation.

For GitHub-native equation rendering, use `docs/WINDOW_REFERENCE.md`.

Quantitative metrics and per-window SVG plots are generated from `docs/window_metrics.json` and `docs/assets/windows/*`.

Window Formula Key

Let $n = 0, \ldots, N-1$, center index $m = (N-1)/2$, and normalized coordinate $x_n = (n-m)/m$.

(W0) Rectangular:

$$w[n] = 1$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W1) Cosine series:

$$w[n] = \sum_{k=0}^{K} a_k \cos\left(\frac{2\pi kn}{N-1}\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W2) Sine/Cosine:

$$w[n] = \sin\left(\frac{\pi n}{N-1}\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W3) Bartlett:

$$w[n] = 1 - \left|\frac{n-m}{m}\right|$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W4) Triangular:

$$w[n] = \max\left(1 - \left|\frac{n-m}{(N+1)/2}\right|, 0\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W5) Bartlett-Hann:

$$x = \frac{n}{N-1} - \frac{1}{2}, \quad w[n] = 0.62 - 0.48|x| + 0.38\cos(2\pi x)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W6) Tukey:

$$x = \frac{n}{N-1}, \quad w[n] = \begin{cases} \frac{1}{2}\left(1+\cos\left(\pi\left(\frac{2x}{\alpha}-1\right)\right)\right), & 0 \le x < \frac{\alpha}{2} \\ 1, & \frac{\alpha}{2} \le x < 1-\frac{\alpha}{2} \\ \frac{1}{2}\left(1+\cos\left(\pi\left(\frac{2x}{\alpha}-\frac{2}{\alpha}+1\right)\right)\right), & 1-\frac{\alpha}{2} \le x \le 1 \end{cases}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

pvx special cases: $\alpha \le 0$ is rectangular and $\alpha \ge 1$ becomes Hann.

(W7) Parzen:

$$u=\left|\frac{2n}{N-1}-1\right|,\quad w[n]=\begin{cases} 1-6u^2+6u^3, & 0\le u\le \frac{1}{2} \ 2(1-u)^3, & \frac{1}{2}1 \end{cases}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W8) Lanczos:

$$w[n] = \mathrm{sinc}\left(\frac{2n}{N-1}-1\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W9) Welch:

$$w[n] = \max\left(1-x_n^2, 0\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W10) Gaussian:

$$w[n] = \exp\left(-\frac{1}{2}\left(\frac{n-m}{\sigma}\right)^2\right), \ \sigma = r_\sigma m$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W11) General Gaussian:

$$w[n] = \exp\left(-\frac{1}{2}\left|\frac{n-m}{\sigma}\right|^{2p}\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W12) Exponential:

$$w[n] = \exp\left(-\frac{|n-m|}{\tau}\right), \ \tau = r_\tau m$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W13) Cauchy:

$$w[n] = \frac{1}{1 + \left(\frac{n-m}{\gamma}\right)^2}, \gamma = r_\gamma m$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W14) Cosine power:

$$w[n] = \sin\left(\frac{\pi n}{N-1}\right)^p$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W15) Hann-Poisson:

$$w[n] = w_{\text{Hann}}[n] \exp\left(-\alpha \frac{|n-m|}{m}\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W16) General Hamming:

$$w[n] = \alpha - (1-\alpha)\cos\left(\frac{2\pi n}{N-1}\right)$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W17) Bohman:

$$x = \left|\frac{2n}{N-1} - 1\right|, \;\; w[n] = (1-x)\cos(\pi x) + \frac{\sin(\pi x)}{\pi}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

(W18) Kaiser:

$$w[n] = \frac{I_0\left(\beta\sqrt{1-r_n^2}\right)}{I_0(\beta)}, \quad r_n = \frac{n-m}{m}$$

where `x` represents the signal (and `x[n]` represents sample index `n`), `t` represents frame index, `k` represents frequency-bin index, `w[n]` represents the analysis window, `H_a` / `H_s` represent analysis/synthesis hop sizes, `\phi` represents phase, `\omega` represents angular frequency, and `\varepsilon` represents a small numerical-stability constant.

Each window name in the table maps to one formula family above plus fixed constants/shape parameters.

Window	Family	Parameters	Formula key	Coherent gain	ENBW (bins)	Scalloping loss (dB)	Main-lobe width (bins)	Peak side-lobe (dB)	Plots	Plain-English behavi
`hann`	Cosine series	`coeffs=(0.5, -0.5)`	W1	0.499756	1.500733	-1.422	4.000	-31.468	time / freq	Cosine-sum tap balanci leakage and resoluti

Window	Family	Parameters	Formula key	Coherent gain	ENBW (bins)	Scalloping loss (dB)	Main-lobe width (bins)	Peak side-lobe (dB)	Plots	Plain-Englisl behavi
`hamming`	Cosine series	`coeffs=(0.54, -0.46)`	W1	0.539775	1.363305	-1.750	4.000	-42.675	time / freq	Cosine-sum tap balanci leakage and resoluti
`blackman`	Cosine series	`coeffs=(0.42, -0.5, 0.08)`	W1	0.419795	1.727601	-1.098	6.000	-58.109	time / freq	Cosine-sum tap balanci leakage and resoluti
`blackmanharris`	Cosine series	`coeffs=(0.35875, -0.48829, 0.14128, -0.01168)`	W1	0.358575	2.005332	-0.825	8.000	-92.011	time / freq	Cosine-sum tap balanci leakage and resoluti
`nuttall`	Cosine series	`coeffs=(0.355768, -0.487396, 0.144232, -0.012604)`	W1	0.355594	2.022220	-0.811	8.000	-93.325	time / freq	Cosine-sum tap balanci leakage and resoluti
`flattop`	Cosine series	`coeffs=(1, -1.93, 1.29, -0.388, 0.0322)`	W1	0.999514	3.772117	-0.016	10.000	-68.311	time / freq	Cosine-sum tap balanci leakage and resoluti
`blackman_nuttall`	Cosine series	`coeffs=(0.363582, -0.489177, 0.1366, -0.0106411)`	W1	0.363405	1.977073	-0.850	8.000	-98.174	time / freq	Cosine-sum tap balanci leakage and resoluti
`exact_blackman`	Cosine series	`coeffs=(0.426591, -0.496561, 0.0768487)`	W1	0.426386	1.694500	-1.149	6.000	-68.236	time / freq	Cosine-sum tap balanci leakage and resoluti
`sine`	Sinusoidal	`none`	W2	0.636309	1.234304	-2.096	3.000	-22.999	time / freq	Raised-sine tap smooth edges v modera main-lo width.
`bartlett`	Triangular	`none`	W3	0.499756	1.333985	-1.822	4.000	-26.523	time / freq	Linear t angular per; sin and fas

Window	Family	Parameters	Formula key	Coherent gain	ENBW (bins)	Scalloping loss (dB)	Main-lobe width (bins)	Peak side-lobe (dB)	Plots	Plain-Englisl behavi
`boxcar`	Rectangular	`none`	W0	1.000000	1.000000	-3.922	2.000	-13.264	time / freq	No tape ing; bes frequer resoluti but stro est leakage
`triangular`	Triangular	`none`	W4	0.500244	1.332683	-1.826	4.000	-26.523	time / freq	Triangu taper va ant with denomi tor $(N+1$.
`bartlett_hann`	Hybrid taper	`fixed coefficients`	W5	0.499756	1.456559	-1.517	4.000	-35.874	time / freq	Blend c Bartlett slope a cosine curvatu
`tukey`	Tukey	`alpha=0.5`	W6	0.749634	1.222819	-2.236	2.656	-15.123	time / freq	Cosine pers at edges v flat cen region.
`tukey_0p1`	Tukey	`alpha=0.1`	W6	0.949536	1.039289	-3.505	2.094	-13.309	time / freq	Cosine pers at edges v flat cen region.
`tukey_0p25`	Tukey	`alpha=0.25`	W6	0.874573	1.102579	-2.960	2.281	-13.601	time / freq	Cosine pers at edges v flat cen region.
`tukey_0p75`	Tukey	`alpha=0.75`	W6	0.624695	1.360664	-1.728	3.188	-19.395	time / freq	Cosine pers at edges v flat cen region.
`tukey_0p9`	Tukey	`alpha=0.9`	W6	0.549731	1.446988	-1.521	3.625	-24.972	time / freq	Cosine pers at edges v flat cen region.
`parzen`	Polynomial	`piecewise cubic`	W7	0.374817	1.918397	-0.897	8.000	-53.046	time / freq	Smooth piecewi cubic w dow wit strong sidelob suppres sion.

Window	Family	Parameters	Formula key	Coherent gain	ENBW (bins)	Scalloping loss (dB)	Main-lobe width (bins)	Peak side-lobe (dB)	Plots	Plain-Englisl behavi
`lanczos`	Sinc	`none`	W8	0.589202	1.299668	-1.889	3.281	-26.405	time / freq	Sinc-ba taper w useful compro mise fo spectra analysis
`welch`	Quadratic	`none`	W9	0.666341	1.200587	-2.223	2.875	-21.295	time / freq	Parabo taper e phasizi center sample
`gaus-sian_0p25`	Gaussian	`sigma_ratio=0.25`	W10	0.313156	2.258144	-0.668	11.406	-87.677	time / freq	Bell-shaped per; sm er sigm gives stronge edge attenua tion.
`gaus-sian_0p35`	Gaussian	`sigma_ratio=0.35`	W10	0.436580	1.626488	-1.268	6.719	-52.077	time / freq	Bell-shaped per; sm er sigm gives stronge edge attenua tion.
`gaus-sian_0p45`	Gaussian	`sigma_ratio=0.45`	W10	0.548949	1.320556	-1.869	4.125	-35.366	time / freq	Bell-shaped per; sm er sigm gives stronge edge attenua tion.
`gaus-sian_0p55`	Gaussian	`sigma_ratio=0.55`	W10	0.641515	1.171861	-2.350	3.000	-28.906	time / freq	Bell-shaped per; sm er sigm gives stronge edge attenua tion.
`gaus-sian_0p65`	Gaussian	`sigma_ratio=0.65`	W10	0.713490	1.097657	-2.705	2.625	-22.575	time / freq	Bell-shaped per; sm er sigm gives stronge edge attenua tion.

Window	Family	Parameters	Formula key	Coherent gain	ENBW (bins)	Scalloping loss (dB)	Main-lobe width (bins)	Peak side-lobe (dB)	Plots	Plain-Englisl behavi
`general_gaussian_1p5_0p35`	Generalized Gaussian	`power=1.5, sigma_ratio=0.35`	W11	0.393587	2.016584	-0.784	5.156	-24.895	time / freq	Adjusta shape e ponent controls shoulde steepne
`general_gaussian_2p0_0p35`	Generalized Gaussian	`power=2, sigma_ratio=0.35`	W11	0.377081	2.230016	-0.633	5.281	-19.717	time / freq	Adjusta shape e ponent controls shoulde steepne
`general_gaussian_3p0_0p35`	Generalized Gaussian	`power=3, sigma_ratio=0.35`	W11	0.364287	2.445593	-0.532	5.469	-16.299	time / freq	Adjusta shape e ponent controls shoulde steepne
`general_gaussian_4p0_0p35`	Generalized Gaussian	`power=4, sigma_ratio=0.35`	W11	0.359267	2.552433	-0.496	5.562	-15.060	time / freq	Adjusta shape e ponent controls shoulde steepne
`exponential_0p25`	Exponential	`tau_ratio=0.25`	W12	0.245310	2.075492	-1.022	15.562	-31.885	time / freq	Symme expone decay away fr center.
`exponential_0p5`	Exponential	`tau_ratio=0.5`	W12	0.432187	1.313323	-2.032	3.625	-19.190	time / freq	Symme expone decay away fr center.
`exponential_1p0`	Exponential	`tau_ratio=1`	W12	0.631991	1.082055	-2.854	2.594	-20.141	time / freq	Symme expone decay away fr center.
`cauchy_0p5`	Cauchy / Lorentzian	`gamma_ratio=0.5`	W13	0.553402	1.229775	-2.209	3.562	-22.733	time / freq	Heavy-tailed ta with slo side de than Gaussi
`cauchy_1p0`	Cauchy / Lorentzian	`gamma_ratio=1`	W13	0.785259	1.041963	-3.108	2.375	-19.034	time / freq	Heavy-tailed ta with slo side de than Gaussi
`cauchy_2p0`	Cauchy / Lorentzian	`gamma_ratio=2`	W13	0.927233	1.004393	-3.648	2.094	-14.689	time / freq	Heavy-tailed ta with slo side de than Gaussi

Window	Family	Parameters	Formula key	Coherent gain	ENBW (bins)	Scalloping loss (dB)	Main-lobe width (bins)	Peak side-lobe (dB)	Plots	Plain-Englis behavi
`cosine_power_2`	Cosine power	`power=2`	W14	0.499756	1.500733	-1.422	4.000	-31.468	time / freq	Raises sine ta to powe higher narrows fective support
`cosine_power_3`	Cosine power	`power=3`	W14	0.424206	1.735739	-1.074	5.000	-39.295	time / freq	Raises sine ta to powe higher narrows fective support
`cosine_power_4`	Cosine power	`power=4`	W14	0.374817	1.945394	-0.862	6.000	-46.741	time / freq	Raises sine ta to powe higher narrows fective support
`hann_poisson_0p5`	Hann-Poisson	`alpha=0.5`	W15	0.432946	1.609944	-1.258	5.188	-35.245	time / freq	Hann m plied by ponenti envelop for stro edge decay.
`hann_poisson_1p0`	Hann-Poisson	`alpha=1`	W15	0.378797	1.734176	-1.112	103.750	-80.017	time / freq	Hann m plied by ponenti envelop for stro edge decay.
`hann_poisson_2p0`	Hann-Poisson	`alpha=2`	W15	0.297878	2.023174	-0.870	164.719	-80.005	time / freq	Hann m plied by ponenti envelop for stro edge decay.
`general_hamming_0p50`	General Hamming	`alpha=0.50`	W16	0.499756	1.500733	-1.422	4.000	-31.468	time / freq	Hammi family v tunable sine weight.
`general_hamming_0p60`	General Hamming	`alpha=0.60`	W16	0.599805	1.222475	-2.179	3.469	-31.600	time / freq	Hammi family v tunable sine weight.
`general_hamming_0p70`	General Hamming	`alpha=0.70`	W16	0.699854	1.091920	-2.762	2.656	-24.078	time / freq	Hammi family v tunable sine weight.

Window	Family	Parameters	Formula key	Coherent gain	ENBW (bins)	Scalloping loss (dB)	Main-lobe width (bins)	Peak side-lobe (dB)	Plots	Plain-English behavi
`general_hamming_0p80`	General Hamming	`alpha=0.80`	W16	0.799902	1.031273	-3.227	2.312	-18.649	time / freq	Hammi family v tunable sine weight.
`bohman`	Bohman	`none`	W17	0.405087	1.786613	-1.022	6.000	-45.997	time / freq	Continu slope w dow wi cosine sine correcti
`cosine`	Sinusoidal	`none`	W2	0.636309	1.234304	-2.096	3.000	-22.999	time / freq	Same i plemen tion as window pvx.
`kaiser`	Kaiser-Bessel	`beta from --kaiser-beta`	W18	0.331708	2.162181	-0.712	9.125	-105.921	time / freq	Adjusta trade-o window ing mod fied Be functior
`rect`	Rectangular	`none`	W0	1.000000	1.000000	-3.922	2.000	-13.264	time / freq	Alias of boxcar pvx.

Complete Plot Gallery (All Windows)

Time-domain and frequency-magnitude plots for every supported window.

Window	Time-domain shape	Magnitude spectrum
`hann`	hann time-domain plot	hann magnitude spectrum
`hamming`	hamming time-domain plot	hamming magnitude spectrum
`blackman`	blackman time-domain plot	blackman magnitude spectrum
`blackmanharris`	blackmanharris time-domain plot	blackmanharris magnitude spectrum
`nuttall`	nuttall time-domain plot	nuttall magnitude spectrum
`flattop`	flattop time-domain plot	flattop magnitude spectrum
`blackman_nuttall`	blackman_nuttall time-domain plot	blackman_nuttall magnitude spectrum
`exact_blackman`	exact_blackman time-domain plot	exact_blackman magnitude spectrum
`sine`	sine time-domain plot	sine magnitude spectrum
`bartlett`	bartlett time-domain plot	bartlett magnitude spectrum
`boxcar`	boxcar time-domain plot	boxcar magnitude spectrum
`triangular`	triangular time-domain plot	triangular magnitude spectrum
`bartlett_hann`	bartlett_hann time-domain plot	bartlett_hann magnitude spectrum

Window	Time-domain shape	Magnitude spectrum
`tukey`	tukey time-domain plot	tukey magnitude spectrum
`tukey_0p1`	tukey_0p1 time-domain plot	tukey_0p1 magnitude spectrum
`tukey_0p25`	tukey_0p25 time-domain plot	tukey_0p25 magnitude spectrum
`tukey_0p75`	tukey_0p75 time-domain plot	tukey_0p75 magnitude spectrum
`tukey_0p9`	tukey_0p9 time-domain plot	tukey_0p9 magnitude spectrum
`parzen`	parzen time-domain plot	parzen magnitude spectrum
`lanczos`	lanczos time-domain plot	lanczos magnitude spectrum
`welch`	welch time-domain plot	welch magnitude spectrum
`gaussian_0p25`	gaussian_0p25 time-domain plot	gaussian_0p25 magnitude spectrum
`gaussian_0p35`	gaussian_0p35 time-domain plot	gaussian_0p35 magnitude spectrum
`gaussian_0p45`	gaussian_0p45 time-domain plot	gaussian_0p45 magnitude spectrum
`gaussian_0p55`	gaussian_0p55 time-domain plot	gaussian_0p55 magnitude spectrum
`gaussian_0p65`	gaussian_0p65 time-domain plot	gaussian_0p65 magnitude spectrum
`general_gaussian_1p5_0p35`	general_gaussian_1p5_0p35 time-domain plot	general_gaussian_1p5_0p35 magnitude spectrum
`general_gaussian_2p0_0p35`	general_gaussian_2p0_0p35 time-domain plot	general_gaussian_2p0_0p35 magnitude spectrum
`general_gaussian_3p0_0p35`	general_gaussian_3p0_0p35 time-domain plot	general_gaussian_3p0_0p35 magnitude spectrum
`general_gaussian_4p0_0p35`	general_gaussian_4p0_0p35 time-domain plot	general_gaussian_4p0_0p35 magnitude spectrum
`exponential_0p25`	exponential_0p25 time-domain plot	exponential_0p25 magnitude spectrum
`exponential_0p5`	exponential_0p5 time-domain plot	exponential_0p5 magnitude spectrum
`exponential_1p0`	exponential_1p0 time-domain plot	exponential_1p0 magnitude spectrum
`cauchy_0p5`	cauchy_0p5 time-domain plot	cauchy_0p5 magnitude spectrum
`cauchy_1p0`	cauchy_1p0 time-domain plot	cauchy_1p0 magnitude spectrum
`cauchy_2p0`	cauchy_2p0 time-domain plot	cauchy_2p0 magnitude spectrum
`cosine_power_2`	cosine_power_2 time-domain plot	cosine_power_2 magnitude spectrum
`cosine_power_3`	cosine_power_3 time-domain plot	cosine_power_3 magnitude spectrum
`cosine_power_4`	cosine_power_4 time-domain plot	cosine_power_4 magnitude spectrum
`hann_poisson_0p5`	hann_poisson_0p5 time-domain plot	hann_poisson_0p5 magnitude spectrum
`hann_poisson_1p0`	hann_poisson_1p0 time-domain plot	hann_poisson_1p0 magnitude spectrum
`hann_poisson_2p0`	hann_poisson_2p0 time-domain plot	hann_poisson_2p0 magnitude spectrum
`general_hamming_0p50`	general_hamming_0p50 time-domain plot	general_hamming_0p50 magnitude spectrum
`general_hamming_0p60`	general_hamming_0p60 time-domain plot	general_hamming_0p60 magnitude spectrum
`general_hamming_0p70`	general_hamming_0p70 time-domain plot	general_hamming_0p70 magnitude spectrum
`general_hamming_0p80`	general_hamming_0p80 time-domain plot	general_hamming_0p80 magnitude spectrum
`bohman`	bohman time-domain plot	bohman magnitude spectrum
`cosine`	cosine time-domain plot	cosine magnitude spectrum
`kaiser`	kaiser time-domain plot	kaiser magnitude spectrum
`rect`	rect time-domain plot	rect magnitude spectrum

References (143 papers)

Source: papers.html

This bibliography collects foundational and directly related literature that informed pvx's phase-vocoder-centric architecture and the broader DSP algorithm roadmap.

Total references: **143** across **8** categories.

Links point to DOI pages, publisher archives, arXiv, standards documents, project docs, or Google Scholar queries where an official landing page can vary by publisher access.

Categories

Phase Vocoder Foundations

Year	Authors	Title	Venue	Link type	Link
2013	M. Dolson	History of the Phase Vocoder	CCRMA Note	`scholar`	Oper
2012	B. Zavalishin	The Art of VA Filter Design (sections on phase and spectral processing)	Book	`scholar`	Oper
2009	N. Moreau	Toolbox for Time-Scale Modification and Pitch-Shifting of Audio Signals	DAFx	`scholar`	Oper
2003	A. Röbel	A New Approach to Transient Processing in the Phase Vocoder	DAFx	`web`	Oper
2002	C. Duxbury; M. Davies; M. Sandler	Improved Time-Scaling of Musical Audio Using Phase Locking at Transients	AES Convention	`scholar`	Oper
2000	J. Bonada	Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio	ICMC	`scholar`	Oper
1999	S. M. Bernsee	Pitch Shifting Using the Fourier Transform	DAFX Workshop Note	`web`	Oper
1999	J. Laroche; M. Dolson	New Phase-Vocoder Techniques for Pitch-Shifting, Harmonizing and Other Exotic Effects	WASPAA	`doi`	Oper
1999	J. Laroche; M. Dolson	Improved Phase Vocoder Time-Scale Modification of Audio	IEEE Trans. Speech and Audio Processing	`doi`	Oper
1998	S. Disch	A New Phase Vocoder Technique for Time-Scale Modification of Audio Signals	DAFx	`scholar`	Oper
1995	M. Puckette	Phase-Locked Vocoder	ICMC	`scholar`	Oper
1990	X. Serra; J. O. Smith	Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition	Computer Music Journal	`publisher`	Oper
1989	R. Bristow-Johnson; M. Bogdanowicz	Phase Vocoder Done Right	AES Preprint	`scholar`	Oper
1986	M. Dolson	The Phase Vocoder: A Tutorial	Computer Music Journal	`publisher`	Oper
1986	R. J. McAulay; T. F. Quatieri	Speech Analysis/Synthesis Based on a Sinusoidal Representation	IEEE Trans. ASSP	`scholar`	Oper
1985	N. Roucos; A. Wilgus	High Quality Time-Scale Modification for Speech	ICASSP	`scholar`	Oper

Year	Authors	Title	Venue	Link type	Link
1984	D. W. Griffin; J. S. Lim	Signal Estimation from Modified Short-Time Fourier Transform	IEEE Trans. ASSP	doi	Oper
1980	M. R. Portnoff	Time-Scale Modification of Speech Based on Short-Time Fourier Analysis	IEEE Trans. ASSP	scholar	Oper
1977	J. B. Allen; L. R. Rabiner	A Unified Approach to Short-Time Fourier Analysis and Synthesis	Proceedings of the IEEE	doi	Oper
1976	M. R. Portnoff	Implementation of the Digital Phase Vocoder Using the Fast Fourier Transform	IEEE Trans. ASSP	scholar	Oper
1966	J. L. Flanagan; R. M. Golden	Phase Vocoder	Bell System Technical Journal	doi	Oper

Time-Scale and Pitch Methods

Year	Authors	Title	Venue	Link type	Link
2016	J. Driedger; M. Müller	A Review of Time-Scale Modification of Music Signals	Applied Sciences	doi	Oper
2015	M. Riess; A. R. Chhetri	Transient Preservation in Time-Scale Modification	WASPAA	scholar	Oper
2014	J. Driedger; T. Pratzlich; M. Muller	Let It Bee - Towards NMF-Inspired Audio Mosaicing	ISMIR	scholar	Oper
2011	M. Le Roux; E. Vincent	Consistent Wiener Filtering for Audio Source Separation and Time-Frequency Processing	IEEE Signal Processing Letters	scholar	Oper
2010	D. S. Hamon; A. Lazarides	Improved WSOLA for Real-Time Time-Scale Modification	AES Convention	scholar	Oper
2010	A. Roebel	A Shape-Invariant Phase Vocoder for Speech Transformation	Interspeech	scholar	Oper
2008	J. Bonada	Wide-Band Harmonic Sinusoidal Modeling for Time-Scale and Pitch-Scale Modification	DAFx	scholar	Oper
2005	A. Röbel; X. Rodet	Efficient Spectral Envelope Estimation and Its Application to Pitch Shifting and Envelope Preservation	DAFx	web	Oper
2004	A. de Cheveigne	Pitch and Time Manipulation of Speech	Tutorial	scholar	Oper
2003	S. M. Bernsee	Time Stretching and Pitch Shifting of Audio Signals - An Overview	Online article	web	Oper
2003	J. Laroche	About this Phasiness Business	DAFx	scholar	Oper
2002	S. Arfib; D. Keiler; U. Zölzer	DAFX: Digital Audio Effects (chapter references on pitch/time processing)	Wiley	scholar	Oper
1993	W. Verhelst; M. Roelands	An Overlap-Add Technique Based on Waveform Similarity (WSOLA) for High Quality Time-Scale Modification of Speech	ICASSP	scholar	Oper
1990	E. Moulines; F. Charpentier	Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones	Speech Communication	doi	Oper

Pitch Detection and Tracking

Year	Authors	Title	Venue	Link type	Link
2018	S. Bock; M. Korzeniowski	piano_transcription - F0 and onset tracking	ISMIR	scholar	Oper
2018	J. W. Kim; J. Salamon; P. Li; J. P. Bello	CREPE: A Convolutional Representation for Pitch Estimation	ICASSP	arxiv	Oper
2017	R. Bittner et al.	Deep Salience Representations for F0 Estimation in Polyphonic Music	ISMIR	arxiv	Oper
2015	B. W. Schuller et al.	Paralinguistics and robust pitch tracking methods	IEEE	scholar	Oper
2014	M. Mauch; S. Dixon	pYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions	ICASSP	doi	Oper
2012	K. Dressler	Sinusoidal Extraction using Frequency-Domain Matching Pursuit	DAFx	scholar	Oper
2008	A. Camacho; J. G. Harris	A Sawtooth Waveform Inspired Pitch Estimator for Speech and Music (SWIPE)	JASA	doi	Oper

Year	Authors	Title	Venue	Link type	Link
2006	A. Klapuri	Multiple Fundamental Frequency Estimation by Harmonicity and Spectral Smoothness	IEEE TASLP	scholar	Oper
2005	M. McLeod; G. Wyvill	A Smarter Way to Find Pitch	ICMC	scholar	Oper
2002	A. de Cheveigné; H. Kawahara	YIN, a Fundamental Frequency Estimator for Speech and Music	JASA	doi	Oper
2001	H. Kawahara; A. de Cheveigne	STRAIGHT, Exploitation of the Other Aspects of Vocal Source Information	ICASSP	scholar	Oper
2000	S. Kum; C. L. Nikias	Robust F0 Estimation in Noise	IEEE	scholar	Oper
1999	S. Ahmadi; H. Spanias	Cepstrum-Based Pitch Detection Using FFT	Conference	scholar	Oper
1995	D. Talkin	A Robust Algorithm for Pitch Tracking (RAPT)	In Speech Coding and Synthesis	scholar	Oper
1993	P. Boersma	Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-Noise Ratio of a Sampled Sound	IFA Proceedings	scholar	Oper
1976	L. R. Rabiner; M. J. Cheng; A. E. Rosenberg; C. A. McGonegal	A Comparative Performance Study of Several Pitch Detection Algorithms	IEEE Trans. ASSP	scholar	Oper
1967	A. M. Noll	Cepstrum Pitch Determination	JASA	scholar	Oper

Time-Frequency and Transform Methods

Year	Authors	Title	Venue	Link type	Link
2019	M. Doring; M. M. Kokuer	Practical NSGT audio applications	DAFx	scholar	Oper
2012	S. Essid; G. Richard	Musical signal analysis with reassignment methods	Book chapter	scholar	Oper
2011	I. Daubechies; J. Lu; H.-T. Wu	Synchrosqueezed Wavelet Transforms	Applied and Computational Harmonic Analysis	doi	Oper
2011	G. Velasco; N. Holighaus; M. Dörfler; T. Grill	Constructing an Invertible Constant-Q Transform with Nonstationary Gabor Frames	DAFx	web	Oper
2010	A. V. Oppenheim; R. W. Schafer	Discrete-Time Signal Processing (STFT and filter-bank chapters)	Pearson	scholar	Oper
2010	C. Schörkhuber; A. Klapuri	Constant-Q Transform Toolbox for Music Processing	SMC	scholar	Oper
2006	A. C. Gilbert; M. J. Strauss	Approximation of signals in sparse Fourier dictionaries	IEEE	scholar	Oper
2002	P. Flandrin; F. Auger; E. Chassande-Mottin	Time-Frequency Reassignment: From Principles to Algorithms	Applications in Time-Frequency Signal Processing	scholar	Oper
2001	K. Grochenig	Foundations of Time-Frequency Analysis	Birkhauser	scholar	Oper
1998	P. Flandrin	Time-Frequency/Time-Scale Analysis	Academic Press	scholar	Oper
1995	F. Auger; P. Flandrin	Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method	IEEE Trans. SP	doi	Oper
1995	D. L. Donoho	De-Noising by Soft-Thresholding	IEEE Trans. Information Theory	scholar	Oper
1993	S. Mallat; Z. Zhang	Matching Pursuits with Time-Frequency Dictionaries	IEEE Trans. Signal Processing	scholar	Oper
1992	R. R. Coifman; M. V. Wickerhauser	Entropy-Based Algorithms for Best Basis Selection	IEEE Trans. IT	doi	Oper
1992	J. C. Brown; M. S. Puckette	An Efficient Algorithm for the Calculation of a Constant Q Transform	JASA	doi	Oper
1991	S. Mann; S. Haykin	The Chirplet Transform: Physical Considerations	IEEE Trans. SP	scholar	Oper
1991	J. C. Brown	Calculation of a Constant Q Spectral Transform	JASA	doi	Oper
1989	L. Cohen	Time-Frequency Distributions - A Review	Proceedings of the IEEE	scholar	Oper
1989	S. Mallat	A Theory for Multiresolution Signal Decomposition: The Wavelet Representation	IEEE Trans. PAMI	doi	Oper

Separation and Decomposition

Year	Authors	Title	Venue	Link type	Link
2019	J. Le Roux; S. Wisdom; H. Erdogan; J. R. Hershey	SDR half-baked or well done?	ICASSP	scholar	Oper
2019	F.-R. Stöter; S. Uhlich; A. Liutkus; Y. Mitsufuji	Open-Unmix - A Reference Implementation for Music Source Separation	Journal of Open Source Software	doi	Oper
2019	F.-R. Stoter; S. Uhlich; A. Liutkus; Y. Mitsufuji	Open-Unmix	ISMIR	scholar	Oper
2019	A. Défossez et al.	Music Source Separation in the Waveform Domain	arXiv	arxiv	Oper
2019	A. Defossez et al.	Music Source Separation in the Waveform Domain	ISMIR	scholar	Oper
2018	N. Takahashi et al.	MMDenseLSTM for source separation	ICASSP	scholar	Oper
2017	A. Jansson et al.	Singing Voice Separation with Deep U-Net Convolutional Networks	ISMIR	arxiv	Oper
2011	N. Ono	Stable and fast update rules for independent low-rank matrix analysis based on auxiliary function technique	WASPAA	scholar	Oper
2011	E. J. Candès; X. Li; Y. Ma; J. Wright	Robust Principal Component Analysis?	JACM	doi	Oper
2011	E. J. Candes; X. Li; Y. Ma; J. Wright	Robust Principal Component Analysis?	Journal of ACM	scholar	Oper
2010	D. Fitzgerald	Harmonic/Percussive Separation Using Median Filtering	DAFx	web	Oper
2009	C. Fevotte; N. Bertin; J.-L. Durrieu	Nonnegative Matrix Factorization with the Itakura-Saito Divergence	Neural Computation	scholar	Oper
2007	T. Virtanen	Monaural Sound Source Separation by Nonnegative Matrix Factorization with Temporal Continuity and Sparseness Criteria	IEEE Trans. Audio, Speech, and Language Processing	scholar	Oper
2007	P. Smaragdis	Convolutive Speech Bases and Their Application to Supervised Speech Separation	IEEE TASLP	scholar	Oper
2001	D. D. Lee; H. S. Seung	Algorithms for Non-negative Matrix Factorization	NIPS	doi	Oper
2000	A. Hyvärinen; E. Oja	Independent Component Analysis: Algorithms and Applications	Neural Networks	doi	Oper
2000	A. Hyvarinen; E. Oja	Independent Component Analysis: Algorithms and Applications	Neural Networks	scholar	Oper
1999	D. D. Lee; H. S. Seung	Learning the Parts of Objects by Non-negative Matrix Factorization	Nature	doi	Oper
1999	A. Hyvärinen	Fast and Robust Fixed-Point Algorithms for Independent Component Analysis	IEEE Trans. Neural Networks	doi	Oper
1995	A. J. Bell; T. J. Sejnowski	An Information-Maximization Approach to Blind Separation and Blind Deconvolution	Neural Computation	scholar	Oper
1994	P. Comon	Independent Component Analysis, A New Concept?	Signal Processing	doi	Oper

Denoising, Dereverberation, and Spatial Audio

Year	Authors	Title	Venue	Link type	Link
2023	EBU	EBU R128: Loudness Normalisation and Permitted Maximum Level of Audio Signals	Recommendation	publisher_pa	Oper
2019	M. H. C. de Gesmundo et al.	ViSQOL v3: An Open Source Production Ready Objective Speech and Audio Metric	QoMEX	scholar	Oper
2019	A. Pandey; D. Wang	A New Framework for CNN-Based Speech Enhancement in the Time Domain	IEEE TASLP	scholar	Oper
2018	N. W. D. Evans et al.	The voicehome and CHiME challenges	IEEE	scholar	Oper
2018	J.-M. Valin	A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement (RNNoise)	MMSP / arXiv	arxiv	Oper

Year	Authors	Title	Venue	Link type	Link
2018	J.-M. Valin	A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement	MMSP	`scholar`	Oper
2016	K. Kinoshita et al.	A summary of the REVERB challenge	EURASIP Journal	`scholar`	Oper
2015	ITU-R	BS.1770-4: Algorithms to Measure Audio Programme Loudness and True-Peak Audio Level	Recommendation	`publisher_`	Oper
2013	P. C. Loizou	Speech Enhancement: Theory and Practice	CRC Press	`scholar`	Oper
2012	A. Hines et al.	VISQOL: An Objective Speech Quality Model	EURASIP	`scholar`	Oper
2012	N. D. Gaubitch; P. A. Naylor	Speech Dereverberation	Springer Handbook	`scholar`	Oper
2012	Y. Yoshioka; T. Nakatani	Generalization of Multi-Channel Linear Prediction Methods for Blind MIMO Impulse Response Shortening	IEEE TASLP	`scholar`	Oper
2012	T. Yoshioka; T. Nakatani	Generalization of Multi-Channel Linear Prediction Methods for Blind MIMO Impulse Response Shortening	IEEE TASLP	`scholar`	Oper
2011	K. K. Paliwal	Importance of phase in speech enhancement	Speech Communication	`scholar`	Oper
2011	C. H. Taal; R. C. Hendriks; R. Heusdens; J. Jensen	An Algorithm for Intelligibility Prediction of Time-Frequency Weighted Noisy Speech (STOI)	IEEE TASLP	`doi`	Oper
2011	C. Taal et al.	An Algorithm for Intelligibility Prediction of Time-Frequency Weighted Noisy Speech	IEEE TASLP	`scholar`	Oper
2010	T. Nakatani; M. Miyoshi; K. Kinoshita	Speech Dereverberation Based on Variance-Normalized Delayed Linear Prediction	IEEE Trans. Audio, Speech, and Language Processing	`doi`	Oper
2001	I. Cohen; B. Berdugo	Speech Enhancement for Non-Stationary Noise Environments	Signal Processing	`scholar`	Oper
2001	A. W. Rix; J. G. Beerends; M. P. Hollier; A. P. Hekstra	Perceptual Evaluation of Speech Quality (PESQ)	ICASSP	`doi`	Oper
2001	A. Rix et al.	Perceptual Evaluation of Speech Quality (PESQ)	ICASSP	`scholar`	Oper
2001	R. Martin	Noise Power Spectral Density Estimation Based on Optimal Smoothing and Minimum Statistics	IEEE Trans. Speech and Audio Processing	`doi`	Oper
1997	S. Pulkki	Virtual Sound Source Positioning Using Vector Base Amplitude Panning	JAES	`scholar`	Oper
1996	P. Scalart; J. V. Filho	Speech Enhancement Based on a Priori Signal to Noise Estimation	ICASSP	`scholar`	Oper
1985	Y. Ephraim; D. Malah	Speech Enhancement Using a Minimum Mean-Square Error Log-Spectral Amplitude Estimator	IEEE Trans. ASSP	`doi`	Oper
1984	Y. Ephraim; D. Malah	Speech Enhancement Using a Minimum Mean-Square Error Short-Time Spectral Amplitude Estimator	IEEE Trans. ASSP	`doi`	Oper
1979	S. F. Boll	Suppression of Acoustic Noise in Speech Using Spectral Subtraction	IEEE Trans. ASSP	`doi`	Oper
1979	S. Boll	Suppression of Acoustic Noise in Speech Using Spectral Subtraction	IEEE TASSP	`scholar`	Oper
1976	C. Knapp; G. Carter	The Generalized Correlation Method for Estimation of Time Delay	IEEE Trans. ASSP	`doi`	Oper
1972	O. L. Frost III	An Algorithm for Linearly Constrained Adaptive Array Processing	Proceedings of the IEEE	`doi`	Oper
1969	J. Capon	High-Resolution Frequency-Wavenumber Spectrum Analysis	Proceedings of the IEEE	`doi`	Oper
1949	N. Wiener	Extrapolation, Interpolation, and Smoothing of Stationary Time Series	Book	`scholar`	Oper

Loudness, Dynamics, and Mastering

Year	Authors	Title	Venue	Link type	Link
2023	EBU	Tech 3343: Production Guidelines	Recommendation	`publisher_`	Oper
2023	EBU	Tech 3342: Loudness Range	Recommendation	`publisher_`	Oper

Year	Authors	Title	Venue	Link type	Link
2023	EBU	Tech 3341: Loudness Metering: EBU Mode	Recommendation	`publisher_`	Open
2023	ITU-R	BS.1770-5: Algorithms to Measure Audio Programme Loudness and True-Peak Audio Level	Recommendation	`publisher_`	Open
2020	S. Stables et al.	A Study of Loudness Normalization in Streaming Services	AES Convention	`scholar`	Open
2018	M. Fenton	Practical Dynamics Processing for Modern Music Production	AES Tutorial	`scholar`	Open
2015	M. Ballou	Handbook for Sound Engineers (Loudness and Dynamics chapters)	CRC Press	`scholar`	Open
2012	D. Giannoulis; M. Massberg; J. Reiss	Digital Dynamic Range Compressor Design	JAES	`scholar`	Open
2011	J. Reiss	Under the Hood of a Dynamic Range Compressor	AES	`scholar`	Open
2007	T. Lund	Loudness and True-Peak in Digital Audio	TC Electronic Whitepaper	`scholar`	Open

ML Audio and Neural Vocoding

Year	Authors	Title	Venue	Link type	Link
2021	S. Wisdom et al.	Differentiable Consistency Constraints for Improved Deep Speech Enhancement	ICASSP	`scholar`	Open
2020	J. Kong et al.	HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis	NeurIPS	`scholar`	Open
2020	J. Engel et al.	DDSP: Differentiable Digital Signal Processing	ICLR	`arxiv`	Open
2019	R. Prenger et al.	WaveGlow: A Flow-based Generative Network for Speech Synthesis	ICASSP	`scholar`	Open
2019	K. Kumar et al.	MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis	NeurIPS	`scholar`	Open
2018	N. Kalchbrenner et al.	Efficient Neural Audio Synthesis	ICML	`scholar`	Open
2018	C. Donahue; J. McAuley; M. Puckette	Adversarial Audio Synthesis	ICLR Workshop	`scholar`	Open
2017	Y. Wang et al.	Tacotron: Towards End-to-End Speech Synthesis	Interspeech	`scholar`	Open
2017	A. Gibiansky et al.	Deep Voice 2: Multi-Speaker Neural Text-to-Speech	NeurIPS Workshop	`scholar`	Open
2016	A. van den Oord et al.	WaveNet: A Generative Model for Raw Audio	ArXiv	`arxiv`	Open

pvx Technical Glossary

Source: glossary.html

Linked glossary for core concepts used throughout pvx algorithms, CLIs, and research docs. Entries include concise definitions plus external references (Wikipedia, standards pages, project docs, and canonical papers).

Total terms: **103** across **10** categories.

Categories

Core DSP Math

Term	Description	External link
Aliasing	Frequency-domain folding caused by undersampling. Unchecked aliasing folds out-of-band content back into the audible band, often creating false partials or metallic artifacts.	Reference
Blackman window	Cosine-sum window with stronger sidelobe suppression. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference
Convolution	Linear filtering operation in time domain equivalent to multiplication in frequency domain. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference
Fast Fourier transform (FFT)	Efficient algorithm for computing the discrete Fourier transform. Its computational efficiency is critical because pvx repeatedly computes FFTs frame-by-frame across channels and files.	Reference
Fourier transform	Transforms a signal from time domain to frequency domain. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference
Hamming window	Cosine-sum window with reduced nearest sidelobe amplitude. Compared with Hann it slightly reduces nearest sidelobes, often at the cost of different amplitude bias characteristics.	Reference
Hann window	Raised cosine window commonly used in STFT analysis. It is a common default because it offers a strong practical balance between leakage control and frequency resolution.	Reference
Inverse FFT (IFFT)	Inverse operation that reconstructs time-domain samples from spectral bins. Accurate inverse transforms are required to reconstruct low-artifact audio after spectral-domain edits.	Reference
Kaiser window	Parametric window controlled by beta parameter. The beta parameter gives explicit control over the width-versus-sidelobe tradeoff, which is useful for task-specific tuning.	Reference
Linear phase	Filter phase response that preserves waveform shape of band-limited signals. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference
Minimum phase	System with minimum group delay for a given magnitude response. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference
Nyquist frequency	Half the sample rate; highest representable frequency in sampled audio. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference
Overlap-add (OLA)	Frame-based reconstruction by summing overlapped windowed segments. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference

Term	Description	External link
Spectral leakage	Energy spread across bins due to finite-duration analysis windows. In pvx workflows this directly affects spectral resolution, leakage behavior, and overlap-add reconstruction accuracy in STFT-based processing.	Reference
Window function	Tapering function applied before spectral analysis to control leakage. Window shape selection is one of the strongest controls on sidelobe suppression versus main-lobe width.	Reference

Phase and Time-Scale DSP

Term	Description	External link
Formant preservation	Preservation of spectral envelope while pitch is changed. This helps avoid chipmunk/monster timbre artifacts when pitch is shifted significantly.	Reference
Granular synthesis	Sound synthesis/modification using short grains. In pvx this concept determines how frames are aligned, phases are propagated, and transients are preserved during time-stretch and pitch operations.	Reference
HPSS	Harmonic-percussive source separation in TF domain. Harmonic and percussive components can then be processed with different settings to preserve punch while shaping sustain.	Reference
Identity phase locking	Phase-locking variant that follows dominant peaks. In pvx this concept determines how frames are aligned, phases are propagated, and transients are preserved during time-stretch and pitch operations.	Reference
LP-PSOLA	Linear-prediction-assisted PSOLA variant. Linear-prediction support can improve source/filter control and reduce timbral drift in speech-like material.	Reference
Phase locking	Locking neighboring bin phases to preserve local waveform structure. Locking strategies reduce local phase incoherence, improving clarity and reducing chorus-like blur in stretched material.	Reference
Phase vocoder	STFT-based time-scale/pitch framework using phase propagation. It underpins most pvx core transforms by decoupling time-scale and pitch behavior through phase-consistent spectral resynthesis.	Reference
TD-PSOLA	Time-domain pitch-synchronous overlap-add technique. It is effective for voiced monophonic content where pitch periods are trackable and edits should remain time-domain natural.	Reference
Time warp map	Piecewise mapping from output time to input time. In pvx this concept determines how frames are aligned, phases are propagated, and transients are preserved during time-stretch and pitch operations.	Reference
Transient preservation	Strategies that protect attacks during spectral time modification. Without transient-aware handling, attacks can smear, so this is central for drums and articulation-critical material.	Reference
WSOLA	Waveform-similarity overlap-add time-scale modification. It is often preferred for moderate speech/music time changes when waveform continuity is more important than heavy spectral editing.	Reference

Pitch and Intonation

Term	Description	External link
Cents	Logarithmic pitch interval unit (1200 cents per octave). Because cents are logarithmic, fixed cent offsets represent musically consistent relative pitch intervals anywhere in the register.	Reference
Equal temperament	Tuning system dividing octave into equal semitone ratios. It is the default modern tuning grid but may be intentionally replaced for historical or microtonal work.	Reference
Harmonic product spectrum	Downsample-and-multiply spectrum for F0 evidence. In pvx retune pipelines this governs pitch-target decisions, microtonal mapping behavior, and the stability of note-to-note correction.	Reference
Just intonation	Tuning based on low-integer frequency ratios. Its pure-ratio intervals can sound more consonant in context but depend strongly on key and harmonic function.	Reference
MIDI Tuning Standard	Specification for non-12TET tuning in MIDI systems. This enables compatible systems to exchange alternate tunings without redefining note identities.	Reference
Portamento	Continuous slide between notes. Retune systems often need slide-aware logic so expressive glides are not flattened into abrupt quantized jumps.	Reference
pYIN	Probabilistic extension of YIN with voicing model. Its probabilistic voicing model can improve contour stability relative to raw framewise estimates.	Reference

Term	Description	External link
RAPT	Robust algorithm for pitch tracking. In pvx retune pipelines this governs pitch-target decisions, microtonal mapping behavior, and the stability of note-to-note correction.	Reference
Scala scale format	Text format for microtonal scales. Using Scala files allows precise import of non-12TET interval maps into repeatable production workflows.	Reference
Subharmonic summation	Weighted subharmonic summation for F0 estimation. In pvx retune pipelines this governs pitch-target decisions, microtonal mapping behavior, and the stability of note-to-note correction.	Reference
SWIPE	Sawtooth-waveform inspired pitch estimator. It emphasizes sawtooth-like harmonic structure, which can improve robustness on certain voiced spectra.	Reference
YIN	Fundamental-frequency estimator using difference-function minima. It is commonly used as a robust baseline detector for monophonic F0 tracks.	Reference

Time-Frequency Transforms

Term	Description	External link
Chirplet transform	Transform using chirped atoms. In pvx transform modules this shapes the tradeoff between time localization, frequency precision, and invertibility of analysis/resynthesis steps.	Reference
Constant-Q transform (CQT)	Transform with logarithmic center frequencies and constant Q. Its logarithmic spacing is especially useful for music analysis where pitch perception is logarithmic.	Reference
Multiresolution analysis	Nested subspace structure for multi-scale representations. In pvx transform modules this shapes the tradeoff between time localization, frequency precision, and invertibility of analysis/resynthesis steps.	Reference
NSGT	Nonstationary Gabor transform with invertible adaptive filter banks. NSGT offers adaptive, invertible filter-bank behavior beyond fixed-resolution STFT framing.	Reference
Reassigned spectrogram	Sharper TF representation by moving energy to local centroids. Reassignment can sharpen time-frequency energy localization for clearer ridge and onset interpretation.	Reference
Synchrosqueezing	Post-processing that concentrates TF energy along ridges. It can concentrate ridge energy for improved instantaneous-frequency interpretation.	Reference
Variable-Q transform (VQT)	Generalized CQT with variable Q and bandwidth behavior. Compared to CQT, VQT broadens control over Q and bandwidth behavior across frequency ranges.	Reference
Wavelet packet	Hierarchical wavelet decomposition of both low/high branches. It supports multi-band decomposition where both approximation and detail branches are recursively analyzed.	Reference

Separation and Decomposition

Term	Description	External link
CP decomposition	Canonical polyadic tensor decomposition. In pvx decomposition workflows this assumption controls source isolation quality, bleed rejection, and the interpretability of separated components.	Reference
Demucs	Deep waveform-domain music source separation architecture. Neural stem separation quality can be high, but behavior depends heavily on training-domain match.	Reference
ICA	Blind source separation based on statistical independence. In pvx decomposition workflows this assumption controls source isolation quality, bleed rejection, and the interpretability of separated components.	Reference
NMF	Nonnegative factorization of matrices into parts-based components. Factor count and constraints strongly influence whether components are musically meaningful or overly fragmented.	Reference
RPCA	Low-rank + sparse decomposition robust to outliers. It is frequently used to separate stable backgrounds from sparse foreground events.	Reference
Tensor decomposition	Higher-order extension of matrix factorization. In pvx decomposition workflows this assumption controls source isolation quality, bleed rejection, and the interpretability of separated components.	Reference
Tucker decomposition	Core tensor with mode matrices factorization. In pvx decomposition workflows this assumption controls source isolation quality, bleed rejection, and the interpretability of separated components.	Reference

Term	Description	External link
U-Net	Encoder-decoder CNN with skip connections. In pvx decomposition workflows this assumption controls source isolation quality, bleed rejection, and the interpretability of separated components.	Reference

Denoising and Restoration

Term	Description	External link
Blind deconvolution	Joint estimation of source and room/filter response. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Declick	Removal of impulsive click artifacts. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Declip	Reconstruction of clipped waveform segments. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Dereverberation	Suppression of late reverberation components. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Diffusion model	Generative model based on iterative denoising processes. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Direct-to-reverberant ratio (DRR)	Ratio of direct sound energy to reverberant energy. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Log-MMSE	Log-spectral MMSE enhancement estimator. Its log-domain objective often better tracks perceptual sensitivity than raw-amplitude objectives.	Reference
Minimum statistics	Noise tracking from low percentile/minimum spectral statistics. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
MMSE-STSA	MMSE short-time spectral amplitude estimator. It estimates clean spectral amplitudes from noisy observations while minimizing expected short-time error.	Reference
RNNoise	Hybrid recurrent neural + DSP speech denoiser. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Spectral subtraction	Subtracts estimated noise spectrum from noisy signal. In pvx restoration chains this controls denoise or dereverb strength versus artifact risk, helping balance intelligibility and texture retention.	Reference
Weighted prediction error (WPE)	Multi-channel dereverb by delayed linear prediction. WPE is a standard dereverberation baseline for suppressing late reverberant tails in multichannel settings.	Reference
Wiener filter	MMSE linear estimator for noisy observations. It is a classic linear MMSE approach and still valuable as a reliable baseline enhancer.	Reference

Dynamics and Mastering

Term	Description	External link
Compander	Combined compressor and expander behavior. In pvx mastering stages this affects loudness compliance, peak containment, and perceived punch versus transparency across final renders.	Reference
Compressor	Dynamic range processor that reduces gain above threshold. Threshold, ratio, attack, and release jointly define how quickly and how strongly level is controlled.	Reference
EBU R128	European loudness standard for production and broadcast. R128 operationalizes BS.1770 loudness measurement into practical broadcast and production targets.	Reference
Expander	Dynamic processor that reduces gain below threshold. It can reduce low-level noise floors and room tail audibility by attenuating content below threshold.	Reference
Hard clipping	Abrupt saturation by truncating waveform beyond threshold. In pvx mastering stages this affects loudness compliance, peak containment, and perceived punch versus transparency across final renders.	Reference
ITU-R BS.1770	International loudness measurement recommendation. Its weighting and gating model is the industry baseline for objective integrated loudness measurement.	Reference

Term	Description	External link
Limiter	High-ratio compressor preventing peaks from exceeding threshold. In mastering chains it is commonly the final safety stage to prevent over-level peaks.	Reference
LUFS	Loudness unit relative to full scale. Targeting LUFS helps keep perceived program loudness consistent across tracks and delivery contexts.	Reference
Soft clipping	Smooth saturation curve reducing harsh distortion. In pvx mastering stages this affects loudness compliance, peak containment, and perceived punch versus transparency across final renders.	Reference
True peak	Peak estimate including inter-sample overs. True-peak metering estimates inter-sample peaks that sample-peak meters can miss.	Reference

Creative Spectral Effects

Term	Description	External link
Amplitude modulation (AM)	Modulating signal amplitude by a control waveform. In pvx creative processing this primarily controls intentional timbral coloration and motion, not transparent corrective behavior.	Reference
Cross synthesis	Combining spectral magnitude and phase or envelopes across signals. Different choices of magnitude/phase or envelope transfer produce very different timbral identities.	Reference
Formant	Resonance region of spectral envelope. In pvx creative processing this primarily controls intentional timbral coloration and motion, not transparent corrective behavior.	Reference
Frequency modulation (FM)	Modulating oscillator frequency by another signal. In pvx creative processing this primarily controls intentional timbral coloration and motion, not transparent corrective behavior.	Reference
LFO	Low-frequency oscillator used for modulation. In pvx creative processing this primarily controls intentional timbral coloration and motion, not transparent corrective behavior.	Reference
Resonator	Filter structure emphasizing a narrow band. In pvx creative processing this primarily controls intentional timbral coloration and motion, not transparent corrective behavior.	Reference
Ring modulation	Multiplication by a carrier to produce sidebands. Sideband generation can create inharmonic or bell-like textures depending on carrier/modulator relationships.	Reference
Spectral freezing	Holding a short-time spectrum and resynthesizing over time. Freeze methods hold a selected spectral snapshot and are useful for drones, pads, and transitions.	Reference

Spatial Audio and Multichannel

Term	Description	External link
Binaural rendering	Headphone rendering preserving spatial localization cues. HRTF quality and listener assumptions strongly affect externalization and localization realism.	Reference
Crosstalk cancellation	Loudspeaker technique reducing opposite-ear leakage. In pvx multichannel workflows this directly influences localization cues, interchannel coherence, and translation to speaker or headphone playback.	Reference
DBAP	Distance-based amplitude panning. DBAP generalizes panning by amplitude weighting from geometric distance rather than simplex triplets.	Reference
Diffuse field	Sound field with many uncorrelated incoming directions. In pvx multichannel workflows this directly influences localization cues, interchannel coherence, and translation to speaker or headphone playback.	Reference
HRTF	Head-related transfer function per source direction. In pvx multichannel workflows this directly influences localization cues, interchannel coherence, and translation to speaker or headphone playback.	Reference
Interaural level difference (ILD)	Level difference cue for localization. In pvx multichannel workflows this directly influences localization cues, interchannel coherence, and translation to speaker or headphone playback.	Reference
Interaural time difference (ITD)	Arrival-time difference cue for localization. In pvx multichannel workflows this directly influences localization cues, interchannel coherence, and translation to speaker or headphone playback.	Reference
Mid/side processing	Stereo representation as sum (mid) and difference (side). In pvx multichannel workflows this directly influences localization cues, interchannel coherence, and translation to speaker or headphone playback.	Reference

Term	Description	External link
Room impulse response (RIR)	Acoustic response from source impulse to receiver. In pvx multichannel workflows this directly influences localization cues, interchannel coherence, and translation to speaker or headphone playback.	Reference
VBAP	Vector-base amplitude panning for loudspeaker arrays. VBAP is widely used for loudspeaker panning with controllable phantom-image placement.	Reference

Analysis and QA

Term	Description	External link
Bayesian optimization	Global optimization of expensive objective functions. It is useful when parameter spaces are expensive to evaluate and objective surfaces are noisy.	Reference
Beat tracking	Estimating temporal pulse positions from audio. In pvx analysis and automation tools this determines feature reliability and how confidently metrics can drive downstream decisions.	Reference
Chord estimation	Inferring harmonic labels from tonal features. In pvx analysis and automation tools this determines feature reliability and how confidently metrics can drive downstream decisions.	Reference
Onset detection	Detection of transient musical events. In pvx analysis and automation tools this determines feature reliability and how confidently metrics can drive downstream decisions.	Reference
PESQ	Perceptual Evaluation of Speech Quality metric. PESQ is speech-focused and should be interpreted alongside task context and listening tests.	Reference
STOI	Short-Time Objective Intelligibility measure. STOI is an intelligibility-oriented metric and does not fully capture timbral preference.	Reference
Structure segmentation	Dividing music into sections like verse/chorus. In pvx analysis and automation tools this determines feature reliability and how confidently metrics can drive downstream decisions.	Reference
ViSQOL	Virtual Speech Quality Objective Listener metric family. ViSQOL provides perceptual similarity estimates and is often used with complementary quality metrics.	Reference

pvx Algorithm Limitations

Source: limitations.html

Limitations guidance for all algorithm groups and algorithm IDs.

Group Summary

Group	Assumptions	Failure modes	When not to use
`analysis_qa_and_automation`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.
`creative_spectral_effects`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`denoise_and_restoration`	Noise/artifacts are distinguishable from desired signal statistics.	Over-reduction can remove detail and create modulation artifacts.	Avoid high reduction settings on sparse acoustic sources without auditioning.
`dereverb_and_room_correction`	Late reverberation is separable from direct content under chosen model.	Speech/music clarity can drop if early reflections are over-suppressed.	Avoid strong dereverb when room character is part of artistic intent.
`dynamics_and_loudness`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are misset.	Avoid applying multiple strong dynamics stages without gain staging checks.
`granular_and_modulation`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.
`pitch_detection_and_tracking`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.
`retune_and_intonation`	Detected notes map cleanly to intended tonal center/scale.	Over-correction can flatten expressive vibrato or slides.	Avoid aggressive correction when preserving natural micro-intonation is required.
`separation_and_decomposition`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or co-modulated sources.
`spatial_and_multichannel`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spectral_time_frequency_transforms`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly non-stationary signals without tuning.
`time_scale_and_pitch_core`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth.	High-ratio stretch can introduce phasiness and blurred transients.	Avoid for extreme percussive-only material when attack realism is critical.

`analysis_qa_and_automation`

Algorithm ID	Assumptions	Failure modes	When not to use
`analysis_qa_and_automation.auto_parameter_tuning_bayesian_optimization`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.
`analysis_qa_and_automation.batch_preset_recommendation_based_on_source_features`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.
`analysis_qa_and_automation.clip_hum_buzz_artifact_detection`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.

Algorithm ID	Assumptions	Failure modes	When not to use
`analysis_qa_and_automation.key_chord_detection`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.
`analysis_qa_and_automation.onset_beat_downbeat_tracking`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.
`analysis_qa_and_automation.pesq_stoi_visqol_quality_metrics`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.
`analysis_qa_and_automation.silence_speech_music_classifiers`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.
`analysis_qa_and_automation.structure_segmentation_verse_chorus_sections`	Feature extraction settings align with domain (speech vs music etc.).	False positives/negatives under domain shift.	Avoid treating single metrics as absolute quality verdicts.

`creative_spectral_effects`

Algorithm ID	Assumptions	Failure modes	When not to use
`creative_spectral_effects.cross_synthesis_vocoder`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`creative_spectral_effects.formant_painting_warping`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`creative_spectral_effects.phase_randomization_textures`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`creative_spectral_effects.resonator_filterbank_morphing`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`creative_spectral_effects.spectral_blur_smear`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`creative_spectral_effects.spectral_contrast_exaggeration`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`creative_spectral_effects.spectral_convolution_effects`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.
`creative_spectral_effects.spectral_freeze_banks`	Spectral manipulations are desired even with timbral coloration.	Can introduce intentional but strong coloration or temporal artifacts.	Avoid for transparent restoration/mastering paths.

`denoise_and_restoration`

Algorithm ID	Assumptions	Failure modes	When not to use
`denoise_and_restoration.declick_decrackle_median_wavelet_interpolation`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.

Algorithm ID	Assumptions	Failure modes	When not to use
`denoise_and_restoration.declip_via_sparse_reconstruction`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.
`denoise_and_restoration.diffusion_based_speech_audio_denoise`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.
`denoise_and_restoration.log_mmse`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.
`denoise_and_restoration.minimum_statistics_noise_tracking`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.
`denoise_and_restoration.mmse_stsa`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.
`denoise_and_restoration.rnnoise_style_denoiser`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.
`denoise_and_restoration.wiener_denoising`	Noise/artifacts are distinguishable from desired signal statistics. Noise model should be representative of observed noise floor.	Over-reduction can remove detail and create modulation artifacts. Mismatched noise estimate leaves residue or damages detail.	Avoid high reduction settings on sparse acoustic sources without auditioning. Avoid static settings on rapidly varying nonstationary noise.

`dereverb_and_room_correction`

Algorithm ID	Assumptions	Failure modes	When not to use
`dereverb_and_room_correction.blind_deconvolution_dereverb`	Late reverberation is separable from direct content under chosen model. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid for intentionally wet effects unless mix preservation is planned.
`dereverb_and_room_correction.drr_guided_dereverb`	Late reverberation is separable from direct content under chosen model. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid for intentionally wet effects unless mix preservation is planned.
`dereverb_and_room_correction.late_reverb_suppression_via_coherence`	Late reverberation is separable from direct content under chosen model. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid for intentionally wet effects unless mix preservation is planned.
`dereverb_and_room_correction.multi_band_adaptive_deverb`	Late reverberation is separable from direct content under chosen model. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid for intentionally wet effects unless mix preservation is planned.

Algorithm ID	Assumptions	Failure modes	When not to use
`dereverb_and_room_correction.neural_dereverb_module`	Late reverberation is separable from direct content under chosen model. Model priors assume training-like signal statistics. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Generalization gaps can produce unstable artifacts. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid fully unattended use on out-of-domain material. Avoid for intentionally wet effects unless mix preservation is planned.
`dereverb_and_room_correction.room_impulse_inverse_filtering`	Late reverberation is separable from direct content under chosen model. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid for intentionally wet effects unless mix preservation is planned.
`dereverb_and_room_correction.spectral_decay_subtraction`	Late reverberation is separable from direct content under chosen model. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid for intentionally wet effects unless mix preservation is planned.
`dereverb_and_room_correction.wpe_dereverberation`	Late reverberation is separable from direct content under chosen model. Reverberation tail is assumed more diffuse than direct content.	Speech/music clarity can drop if early reflections are over-suppressed. Over-suppression can thin tonal body and ambience.	Avoid strong dereverb when room character is part of artistic intent. Avoid for intentionally wet effects unless mix preservation is planned.

`dynamics_and_loudness`

Algorithm ID	Assumptions	Failure modes	When not to use
`dynamics_and_loudness.ebu_r128_normalization`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.
`dynamics_and_loudness.itu_bs_1770_loudness_measurement_gating`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.
`dynamics_and_loudness.lufs_target_mastering_chain`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.
`dynamics_and_loudness.multi_band_compression`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.
`dynamics_and_loudness.spectral_dynamics_bin_wise_compressor_expander`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.
`dynamics_and_loudness.transient_shaping`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.
`dynamics_and_loudness.true_peak_limiting`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.
`dynamics_and_loudness.upward_compression`	Program dynamics fit compressor/limiter time constants and thresholds.	Pumping, breathing, or overs if thresholds and release are mis-set.	Avoid applying multiple strong dynamics stages without gain staging checks.

`granular_and_modulation`

Algorithm ID	Assumptions	Failure modes	When not to use
`granular_and_modulation.am_fm_ring_modulation_blocks`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.
`granular_and_modulation.envelope_followed_modulation_routing`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.

Algorithm ID	Assumptions	Failure modes	When not to use
`granular_and_modulation.formant_lfo_modulation`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.
`granular_and_modulation.freeze_grain_morphing`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.
`granular_and_modulation.grain_cloud_pitch_textures`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.
`granular_and_modulation.granular_time_stretch_engine`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.
`granular_and_modulation.rhythmic_gate_stutter_quantizer`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.
`granular_and_modulation.spectral_tremolo`	Grain and modulation rates are musically matched to source texture.	Incoherent grain scheduling can produce choppiness or blur.	Avoid dense granular settings on speech intelligibility-critical content.

`pitch_detection_and_tracking`

Algorithm ID	Assumptions	Failure modes	When not to use
`pitch_detection_and_tracking.crepe_style_neural_f0`	F0 evidence is strong in the selected analysis band and frame size. Model priors assume training-like signal statistics.	Octave errors and voicing flips under heavy noise/polyphony. Generalization gaps can produce unstable artifacts.	Avoid as the sole control signal for dense polyphonic mixtures. Avoid fully unattended use on out-of-domain material.
`pitch_detection_and_tracking.harmonic_product_spectrum_hps`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.
`pitch_detection_and_tracking.pyin`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.
`pitch_detection_and_tracking.rapt`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.
`pitch_detection_and_tracking.subharmonic_summation`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.
`pitch_detection_and_tracking.swipe`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.
`pitch_detection_and_tracking.viterbi_smoothed_pitch_contour_tracking`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.
`pitch_detection_and_tracking.yin`	F0 evidence is strong in the selected analysis band and frame size.	Octave errors and voicing flips under heavy noise/polyphony.	Avoid as the sole control signal for dense polyphonic mixtures.

`retune_and_intonation`

Algorithm ID	Assumptions	Failure modes	When not to use
`retune_and_intonation.adaptive_intonation_context_sensitive_intervals`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.
`retune_and_intonation.chord_aware_retuning`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.
`retune_and_intonation.just_intonation_mapping_per_key_center`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.
`retune_and_intonation.key_aware_retuning_with_confidence_weighting`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.
`retune_and_intonation.portamento_aware_retune_curves`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.
`retune_and_intonation.scala_mts_scale_import_and_quantization`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.
`retune_and_intonation.time_varying_cents_maps`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.
`retune_and_intonation.vibrato_preserving_correction`	Detected notes map cleanly to intended tonal center/scale. Pitch trajectory estimates should be continuous enough for retuning.	Over-correction can flatten expressive vibrato or slides. Fast F0 jumps can cause audible stepping.	Avoid aggressive correction when preserving natural micro-intonation is required. Avoid high-strength retune on breath/noise segments.

`separation_and_decomposition`

Algorithm ID	Assumptions	Failure modes	When not to use
`separation_and_decomposition.demucs_style_stem_separation_backend`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or comodulated sources.
`separation_and_decomposition.ica_bss_for_multichannel_stems`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or comodulated sources.
`separation_and_decomposition.nmf_decomposition`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or comodulated sources.
`separation_and_decomposition.probabilistic_latent_component_separation`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or comodulated sources.
`separation_and_decomposition.rpca_hpss`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or comodulated sources.
`separation_and_decomposition.sinusoidal_residual_transient_decomposition`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or comodulated sources.
`separation_and_decomposition.tensor_decomposition_cp_tucker`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or comodulated sources.

Algorithm ID	Assumptions	Failure modes	When not to use
`separation_and_decomposition.u_net_vocal_accompaniment_split`	Sources have partially separable spectral or statistical structure.	Component bleeding and musical noise under overlap or model mismatch.	Avoid expecting perfect stems from strongly correlated or co-modulated sources.

`spatial_and_multichannel`

Algorithm ID	Assumptions	Failure modes	When not to use
`spatial_and_multichannel.binaural_itd_ild_synthesis`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.binaural_motion_trajectory_designer`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.coherence_based_dereverb_multichannel`	Channel geometry/order and timing metadata are correct. Reverberation tail is assumed more diffuse than direct content.	Spatial collapse, combing, or localization bias from misalignment. Over-suppression can thin tonal body and ambience.	Avoid blind spatial processing when channel order/calibration is unknown. Avoid for intentionally wet effects unless mix preservation is planned.
`spatial_and_multichannel.cross_channel_click_pop_repair`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.dbap_distance_based_amplitude_panning`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.decorrelated_reverb_upmix`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.microphone_array_calibration_tones`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.multichannel_noise_psd_tracking`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.multichannel_wiener_postfilter`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.phase_aligned_mid_side_field_rotation`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.phase_consistent_multichannel_denoise`	Channel geometry/order and timing metadata are correct. Noise model should be representative of observed noise floor.	Spatial collapse, combing, or localization bias from misalignment. Mismatched noise estimate leaves residue or damages detail.	Avoid blind spatial processing when channel order/calibration is unknown. Avoid static settings on rapidly varying nonstationary noise.
`spatial_and_multichannel.pvx_directional_spectral_warp`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.pvx_interaural_coherence_shaping`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.pvx_interchannel_phase_locking`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.pvx_multichannel_time_alignment`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.

Algorithm ID	Assumptions	Failure modes	When not to use
`spatial_and_multichannel.pvx_spatial_freeze_and_trajectory`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.pvx_spatial_transient_preservation`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.rotating_speaker_doppler_field`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.spatial_freeze_resynthesis`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.spectral_spatial_granulator`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.stereo_width_frequency_dependent_control`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.stochastic_spatial_diffusion_cloud`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.transaural_crosstalk_cancellation`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.
`spatial_and_multichannel.vbap_adaptive_panning`	Channel geometry/order and timing metadata are correct.	Spatial collapse, combing, or localization bias from misalignment.	Avoid blind spatial processing when channel order/calibration is unknown.

`spectral_time_frequency_transforms`

Algorithm ID	Assumptions	Failure modes	When not to use
`spectral_time_frequency_transforms.chirplet_transform_analysis`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.
`spectral_time_frequency_transforms.constant_q_transform_cqt_processing`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.
`spectral_time_frequency_transforms.multi_window_stft_fusion`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.
`spectral_time_frequency_transforms.nsgt_based_processing`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.
`spectral_time_frequency_transforms.reassigned_spectrogram_methods`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.
`spectral_time_frequency_transforms.synchrosqueezed_stft`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.
`spectral_time_frequency_transforms.variable_q_transform_vqt`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.
`spectral_time_frequency_transforms.wavelet_packet_processing`	Transform parameterization matches target time-frequency structure.	Incorrect parameterization can smear events or over-fragment spectra.	Avoid default settings for highly nonstationary signals without tuning.

`time_scale_and_pitch_core`

Algorithm ID	Assumptions	Failure modes	When not to use
`time_scale_and_pitch_core.beat_synchronous_time_warping`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth.	High-ratio stretch can introduce phasiness and blurred transients.	Avoid for extreme percussive-only material when attack realism is critical.
`time_scale_and_pitch_core.harmonic_percussive_split_tsm`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth.	High-ratio stretch can introduce phasiness and blurred transients.	Avoid for extreme percussive-only material when attack realism is critical.
`time_scale_and_pitch_core.lp_psola`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth.	High-ratio stretch can introduce phasiness and blurred transients.	Avoid for extreme percussive-only material when attack realism is critical.
`time_scale_and_pitch_core.multi_resolution_phase_vocoder`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth. Phase continuity assumptions hold best for moderate stretch ratios.	High-ratio stretch can introduce phasiness and blurred transients. Extreme settings increase phasiness/transient blur risk.	Avoid for extreme percussive-only material when attack realism is critical. Avoid very large stretch+pitch shifts without transient controls.
`time_scale_and_pitch_core.nonlinear_time_maps`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth.	High-ratio stretch can introduce phasiness and blurred transients.	Avoid for extreme percussive-only material when attack realism is critical.
`time_scale_and_pitch_core.td_psola`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth.	High-ratio stretch can introduce phasiness and blurred transients.	Avoid for extreme percussive-only material when attack realism is critical.
`time_scale_and_pitch_core.wsola_waveform_similarity_overlap_add`	Frames are locally quasi-stationary and harmonic evolution is reasonably smooth.	High-ratio stretch can introduce phasiness and blurred transients.	Avoid for extreme percussive-only material when attack realism is critical.

pvx Benchmark Report

Source: benchmarks.html

Reproducible STFT/ISTFT benchmark summary.

```
python3 scripts_generate_docs_extras.py --run-benchmarks
```

Spec: sample_rate=48000 Hz, duration=4.0 s

Host: macOS-15.1.1-arm64-arm-64bit-Mach-O | machine=arm64 | python=3.14.3

Backend	Status	Elapsed (ms)	Peak host memory (MB)	SNR vs input (dB)	Spectral dist vs input (dB)	SNR vs CPU (dB)	Spectral dist vs CPU (dB)	Notes
`cpu`	ok	20.356	12.02	160.5928	0.0	n/a	n/a	
`cuda`	unavailable	n/a	n/a	n/a	n/a	n/a	n/a	CUDA mode requires CuPy. Install a matching `cupy-cudaXXx` package.
`apple_silicon_native_cpu`	ok	20.276	12.02	160.5928	0.0	160.5928	0.0	

pvx Pipeline Cookbook

Source: cookbook.html

Pipeline cookbook for practical one-liners, including Unix pipes and mastering chains.

Markdown source: `docs/PIPELINE_COOKBOOK.md`.

Phase-vocoder core

Workflow	Command	Why
Moderate vocal stretch with formant preservation	`python3 pvxvoc.py vocal.wav --time-stretch 1.15 --pitch-mode formant-preserving --output-dir out --suffix _pv`	Retains speech-like vowel envelope while stretching timing.
Independent cents retune	`python3 pvxvoc.py lead.wav --pitch-shift-cents -23 --time-stretch 1.0 --output-dir out --suffix _cents`	Applies precise microtonal offset without tempo change.
Extreme stretch with multistage strategy	`python3 pvxvoc.py ambience.wav --target-duration 600 --ambient-preset --n-fft 16384 --win-length 16384 --hop-size 2048 --window kaiser --kaiser-beta 18 --output-dir out --suffix _ambient600x`	PaulStretch-style ambient profile for very large ratios using stochastic phase and onset time-credit controls.
Ultra-smooth speech stretch (600x)	`python3 pvxvoc.py speech.wav --target-duration 600 --stretch-mode standard --phase-engine propagate --phase-locking identity --n-fft 8192 --win-length 8192 --hop-size 256 --window hann --normalize peak --peak-dbfs -1 --compressor-threshold-db -30 --compressor-ratio 2.0 --compressor-attack-ms 25 --compressor-release-ms 250 --compressor-makeup-db 4 --limiter-threshold 0.95 --output-dir out --suffix _speech600x`	Prefers continuity and intelligibility over texture animation; avoids choppy stochastic artifacts on speech sources.
Auto-profile plan preview	`python3 pvxvoc.py input.wav --auto-profile --auto-transform --explain-plan`	Prints the resolved profile/config plan before long renders.
Multi-resolution fusion stretch	`python3 pvxvoc.py input.wav --multires-fusion --multires-ffts 1024,2048,4096 --multires-weights 0.2,0.35,0.45 --time-stretch 1.4 --output-dir out --suffix _multires`	Blends several FFT scales to reduce single-resolution bias on complex program material.
Checkpointed long render with manifest	`python3 pvxvoc.py long.wav --time-stretch 12 --auto-segment-seconds 0.5 --checkpoint-dir checkpoints --manifest-json reports/run_manifest.json --output-dir out --suffix _long`	Caches segment renders for resume workflows and writes run metadata for reproducibility.

Transform selection

Workflow	Command	Why
Default production backend (FFT + transient protection)	`python3 pvxvoc.py mix.wav --transform fft --time-stretch 1.07 --transient-preserve --phase-locking identity --output-dir out --suffix _fft`	Use when you need the fastest and most stable general-purpose phase-vocoder path.
Reference Fourier baseline using explicit DFT mode	`python3 pvxvoc.py tone_sweep.wav --transform dft --time-stretch 1.00 --pitch-shift-semitones 0 --output-dir out --suffix _dft_ref`	Useful for parity checks and controlled transform-comparison experiments.
Prime-size frame experiment with CZT backend	`python3 pvxvoc.py archival_take.wav --transform czt --n-fft 1531 --win-length 1531 --hop-size 382 --time-stretch 1.03 --output-dir out --suffix _czt`	Alternative numerical path for awkward/prime frame sizes when validating edge cases.
DCT timbral compaction for smooth harmonic material	`python3 pvxvoc.py strings.wav --transform dct --pitch-shift-cents -18 --soft-clip-level 0.95 --output-dir out --suffix _dct`	Real-basis coefficients can emphasize envelope-like structure for creative reshaping.

Workflow	Command	Why
DST odd-symmetry color pass	`python3 pvxvoc.py snare_loop.wav --transform dst --time-stretch 0.92 --phase-locking off --output-dir out --suffix _dst`	Provides an alternate real-basis artifact profile useful for creative percussive processing.
Hartley real-basis exploratory render	`python3 pvxvoc.py synth_pad.wav --transform hartley --time-stretch 1.30 --pitch-shift-semitones 3 --output-dir out --suffix _hartley`	Compares Hartley-domain behavior against complex FFT phase-vocoder output.
A/B sweep of transform backends from shell loop	`for t in fft dft czt dct dst hartley; do python3 pvxvoc.py voice.wav --transform "$t" --time-stretch 1.1 --output-dir out --suffix "_$t"; done`	Fast listening workflow for selecting the least-artifact transform on your source.

Microtonal

Workflow	Command	Why
Custom cents map retune	`python3 pvxretune.py vox.wav --root 60 --scale-cents 0,90,204,294,408,498,612,702,816,906,1020,1110 --strength 0.8 --output-dir out`	Maps incoming notes to a custom 12-degree microtonal scale.
Conform CSV with per-segment ratios	`python3 pvxconform.py solo.wav map_conform.csv --pitch-mode ratio --output-dir out --suffix _conform`	Applies timeline-specific time and pitch trajectories from CSV.

Pipelines

Workflow	Command	Why
Time-stretch -> denoise -> dereverb in one pipe	`python3 pvxvoc.py input.wav --time-stretch 1.25 --stdout | python3 pvxdenoise.py - --reduction-db 10 --stdout | python3 pvxdeverb.py - --strength 0.45 --output-dir out --suffix _clean`	Single-pass CLI chain for serial DSP in Unix pipes.
Morph -> formant -> unison	`python3 pvxmorph.py a.wav b.wav -o - | python3 pvxformant.py - --mode preserve --stdout | python3 pvxunison.py - --voices 5 --detune-cents 8 --output-dir out --suffix _morph_stack`	Builds a richer timbre chain with no intermediate files.
Pitch-follow sidechain map (A controls B)	`python3 HPS-pitch-track.py A.wav | python3 pvxvoc.py B.wav --pitch-follow-stdin --pitch-conf-min 0.75 --pitch-lowconf-mode hold --time-stretch-factor 1.0 --output output.wav`	Tracks F0 contour from source A and applies it as a dynamic pitch-ratio control map on source B.

Mastering

Workflow	Command	Why
Integrated loudness targeting with limiter	`python3 pvxvoc.py mix.wav --time-stretch 1.0 --target-lufs -14 --compressor-threshold-db -20 --compressor-ratio 3 --limiter-threshold 0.98 --output-dir out --suffix _master`	Combines dynamics and loudness controls in shared mastering chain.
Soft clip and hard safety ceiling	`python3 pvxharmonize.py bus.wav --intervals 0,7,12 --mix 0.35 --soft-clip-level 0.92 --soft-clip-type tanh --hard-clip-level 0.99 --output-dir out`	Adds saturation while enforcing a strict final peak ceiling.

Batch

Workflow	Command	Why
Batch stretch over folder	`python3 pvxvoc.py stems/*.wav --time-stretch 1.08 --output-dir out/stems --overwrite`	Applies consistent transform to many files with one command.
Dry-run output validation	`python3 pvxdenoise.py takes/*.wav --reduction-db 8 --dry-run --output-dir out/preview`	Checks filename resolution and collisions without writing audio.

Automation

Workflow	Command	Why
A/B report generation	`python3 scripts_ab_compare.py --input mix.wav --a-args "--time-stretch 1.1 --transform fft" --b-args "--time-stretch 1.1 --transform dct" --out-dir reports/ab --name fft_vs_dct`	Creates JSON/Markdown objective reports for fast algorithm and parameter comparisons.
Benchmark matrix sweep	`python3 scripts_benchmark_matrix.py --input mix.wav --transforms fft,dft,dct --windows hann,kaiser --n-ffts 1024,2048 --devices cpu --out-dir reports/bench`	Produces reproducible CSV/JSON runtime matrices across transform/window/FFT combinations.
Quality regression check	`python3 scripts_quality_regression.py --input mix.wav --output out/reg.wav --render-args "--time-stretch 1.2 --transform fft" --baseline-json reports/baseline.json --report-json reports/regression.json`	Compares current renders against baseline objective metrics with configurable tolerances.

Spatial

Workflow	Command	Why
VBAP adaptive panning via algorithm dispatcher	`python3 -m pvx.algorithms.spatial_and_multichannel.imaging_and_panning.vbap_adaptive_panning input.wav --output-channels 6 --azimuth-deg 35 --width 0.8 --output out/vbap.wav`	Demonstrates algorithm-level spatial module invocation.

Analysis/QA

Workflow	Command	Why
Quality metrics on processed speech	`python3 -m pvx.algorithms.analysis_qa_and_automation.pesq_stoi_visqol_quality_metrics clean.wav noisy.wav --output out/qa.json`	Collects objective quality indicators for regression tracking.

pvx Command-Line Interface (CLI) Flag Index

Source: cli_flags.html

Total unique long flags: **183**

Source index: `docs/CLI_FLAGS_REFERENCE.md` and `docs/cli_flags_reference.json`.

hps_pitch_track.py

Flag	Required	Default	Choices	Action	Description	Source
`--backend`	False	`auto`	`auto, pyin, acf`		Pitch backend (default: auto -> pyin if available, else acf)	`src/pvx/cli/hps_pitch_track.py`
`--confidence-floor`	False	`0.0`			Set confidence below this floor to 0.0 (default: 0.0).	`src/pvx/cli/hps_pitch_track.py`
`--fmax`	False	`1200.0`			Maximum F0 in Hz (default: 1200)	`src/pvx/cli/hps_pitch_track.py`
`--fmin`	False	`50.0`			Minimum F0 in Hz (default: 50)	`src/pvx/cli/hps_pitch_track.py`
`--frame-length`	False	`2048`			Frame length in samples (default: 2048)	`src/pvx/cli/hps_pitch_track.py`
`--hop-size`	False	`256`			Hop size in samples (default: 256)	`src/pvx/cli/hps_pitch_track.py`
`--output`	False				Output CSV path (default: '-' for stdout)	`src/pvx/cli/hps_pitch_track.py`
`--ratio-max`	False	`4.0`			Upper clamp for emitted pitch_ratio (default: 4.0).	`src/pvx/cli/hps_pitch_track.py`
`--ratio-min`	False	`0.25`			Lower clamp for emitted pitch_ratio (default: 0.25).	`src/pvx/cli/hps_pitch_track.py`
`--ratio-reference`	False	`median`	`median, mean, first, hz`		Reference for emitted pitch_ratio values (default: median voiced f0).	`src/pvx/cli/hps_pitch_track.py`
`--reference-hz`	False				Reference frequency in Hz when --ratio-reference hz.	`src/pvx/cli/hps_pitch_track.py`
`--smooth-frames`	False	`5`			Smoothing window for pitch_ratio frames (default: 5).	`src/pvx/cli/hps_pitch_track.py`
`--stretch`	False	`1.0`			Emit constant stretch column value (default: 1.0).	`src/pvx/cli/hps_pitch_track.py`

pvx.py

Flag	Required	Default	Choices	Action	Description	Source
`--chunk-seconds`	False	`0.25`			Chunk/segment duration for `--auto-segment-seconds` (default: 0.25)	`src/pvx/cli/pvx.py`
`--context-ms`	False				Optional stateful context window in milliseconds (default: auto from window/hop)	`src/pvx/cli/pvx.py`
`--crossfade-ms`	False	`0.0`			Crossfade used for segment assembly in milliseconds (default: 0.0)	`src/pvx/cli/pvx.py`
`--example`	False			`store_true`	Print a copy-paste chain example and exit	`src/pvx/cli/pvx.py`

Flag	Required	Default	Choices	Action	Description	Source
`--keep-intermediate`	False			`store_true`	Keep intermediate stage files after successful completion	`src/pvx/cli/pvx.py`
`--mode`	False	`stateful`	`stateful, wrapper`		Stream engine: stateful chunk processor (default) or wrapper compatibility mode	`src/pvx/cli/pvx.py`
`--out`	True				Final output path (or '-')	`src/pvx/cli/pvx.py`
`--output`	True				Final output path (or '-')	`src/pvx/cli/pvx.py`
`--pipeline`	True				Pipeline string with stages separated by '|'. Example: "voc --stretch 1.2 | formant --mode preserve"	`src/pvx/cli/pvx.py`
`--work-dir`	False				Optional directory for intermediate stage files	`src/pvx/cli/pvx.py`

pvxconform.py

Flag	Required	Default	Choices	Action	Description	Source
`--crossfade-ms`	False	`8.0`			Segment crossfade in milliseconds	`src/pvx/cli/pvxconform.py`
`--map`	True				CSV map path	`src/pvx/cli/pvxconform.py`
`--resample-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxconform.py`

pvxdenoise.py

Flag	Required	Default	Choices	Action	Description	Source
`--floor`	False	`0.1`			Noise floor multiplier	`src/pvx/cli/pvxdenoise.py`
`--noise-file`	False				Optional external noise reference	`src/pvx/cli/pvxdenoise.py`
`--noise-seconds`	False	`0.35`			Noise profile duration from start	`src/pvx/cli/pvxdenoise.py`
`--reduction-db`	False	`12.0`			Reduction strength in dB	`src/pvx/cli/pvxdenoise.py`
`--smooth`	False	`5`			Temporal smoothing frames	`src/pvx/cli/pvxdenoise.py`

pvxdeverb.py

Flag	Required	Default	Choices	Action	Description	Source
`--decay`	False	`0.92`			Tail memory decay 0..1	`src/pvx/cli/pvxdeverb.py`
`--floor`	False	`0.12`			Per-bin floor multiplier	`src/pvx/cli/pvxdeverb.py`
`--strength`	False	`0.45`			Tail suppression strength 0..1	`src/pvx/cli/pvxdeverb.py`

pvxformant.py

Flag	Required	Default	Choices	Action	Description	Source
`--formant-lifter`	False	`32`				`src/pvx/cli/pvxformant.py`
`--formant-max-gain-db`	False	`12.0`				`src/pvx/cli/pvxformant.py`

Flag	Required	Default	Choices	Action	Description	Source
`--formant-shift-ratio`	False	`1.0`			Formant ratio (>1 up, <1 down)	`src/pvx/cli/pvxformant.py`
`--mode`	False	`shift`	`shift, preserve`			`src/pvx/cli/pvxformant.py`
`--pitch-shift-cents`	False	`0.0`			Additional microtonal pitch shift in cents before formant stage	`src/pvx/cli/pvxformant.py`
`--pitch-shift-semitones`	False	`0.0`			Optional pitch shift before formant stage	`src/pvx/cli/pvxformant.py`
`--resample-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxformant.py`

pvxfreeze.py

Flag	Required	Default	Choices	Action	Description	Source
`--duration`	False	`3.0`			Output freeze duration in seconds	`src/pvx/cli/pvxfreeze.py`
`--freeze-time`	False	`0.2`			Freeze anchor time in seconds	`src/pvx/cli/pvxfreeze.py`
`--random-phase`	False			`store_true`	Add subtle phase randomization per frame	`src/pvx/cli/pvxfreeze.py`

pvxharmonize.py

Flag	Required	Default	Choices	Action	Description	Source
`--force-stereo`	False			`store_true`	Mix result as stereo with panning	`src/pvx/cli/pvxharmonize.py`
`--gains`	False				Optional comma-separated linear gain per voice	`src/pvx/cli/pvxharmonize.py`
`--intervals`	False	`0,4,7`			Comma-separated semitone intervals per voice (supports fractional values)	`src/pvx/cli/pvxharmonize.py`
`--intervals-cents`	False				Optional cents offsets per voice, added to --intervals (e.g. 0,14,-12)	`src/pvx/cli/pvxharmonize.py`
`--pans`	False				Optional comma-separated pan per voice [-1..1]	`src/pvx/cli/pvxharmonize.py`
`--resample-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxharmonize.py`

pvxlayer.py

Flag	Required	Default	Choices	Action	Description	Source
`--harmonic-gain`	False	`1.0`				`src/pvx/cli/pvxlayer.py`
`--harmonic-kernel`	False	`31`				`src/pvx/cli/pvxlayer.py`
`--harmonic-pitch-cents`	False	`0.0`				`src/pvx/cli/pvxlayer.py`
`--harmonic-pitch-semitones`	False	`0.0`				`src/pvx/cli/pvxlayer.py`
`--harmonic-stretch`	False	`1.0`				`src/pvx/cli/pvxlayer.py`
`--percussive-gain`	False	`1.0`				`src/pvx/cli/pvxlayer.py`
`--percussive-kernel`	False	`31`				`src/pvx/cli/pvxlayer.py`
`--percussive-pitch-cents`	False	`0.0`				`src/pvx/cli/pvxlayer.py`

Flag	Required	Default	Choices	Action	Description	Source
`--percussive-pitch-semitones`	False	`0.0`				`src/pvx/cli/pvxlayer.py`
`--percussive-stretch`	False	`1.0`				`src/pvx/cli/pvxlayer.py`
`--resample-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxlayer.py`

pvxmorph.py

Flag	Required	Default	Choices	Action	Description	Source
`--alpha`	False	`0.5`			Morph amount 0..1 (0=A, 1=B)	`src/pvx/cli/pvx-morph.py`
`--output`	False				Output file path	`src/pvx/cli/pvx-morph.py`
`--output-format`	False				Output extension/format; for --stdout defaults to wav	`src/pvx/cli/pvx-morph.py`
`--overwrite`	False			`store_true`		`src/pvx/cli/pvx-morph.py`
`--stdout`	False			`store_true`	Write processed audio to stdout stream (for piping); equivalent to -o -	`src/pvx/cli/pvx-morph.py`

pvxretune.py

Flag	Required	Default	Choices	Action	Description	Source
`--chunk-ms`	False	`80.0`			Analysis/process chunk duration in ms	`src/pvx/cli/pvxre-tune.py`
`--f0-max`	False	`1200.0`				`src/pvx/cli/pvxre-tune.py`
`--f0-min`	False	`60.0`				`src/pvx/cli/pvxre-tune.py`
`--over-lap-ms`	False	`20.0`			Chunk overlap in ms	`src/pvx/cli/pvxre-tune.py`
`--resam-ple-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxre-tune.py`
`--root`	False	`C`			Scale root note (C,C#,D,...,B)	`src/pvx/cli/pvxre-tune.py`
`--scale`	False	`chro-matic`			Named scale for 12-TET quantization	`src/pvx/cli/pvxre-tune.py`
`--scale-cents`	False				Optional comma-separated microtonal scale degrees in cents within one octave, relative to --root (example: 0,90,204,294,408,498,612,702,816,906,1020,1110)	`src/pvx/cli/pvxre-tune.py`
`--strength`	False	`0.85`			Correction strength 0..1	`src/pvx/cli/pvxre-tune.py`

pvxtransient.py

Flag	Required	Default	Choices	Action	Description	Source
`--pitch-shift-cents`	False				Optional microtonal pitch shift in cents (added to --pitch-shift-semitones)	`src/pvx/cli/pvxtran-sient.py`

Flag	Required	Default	Choices	Action	Description	Source
`--pitch-shift-ratio`	False				Pitch ratio override. Accepts decimals (1.5), integer ratios (3/2), and expressions (2^(1/12)).	`src/pvx/cli/pvxtransient.py`
`--pitch-shift-semitones`	False	`0.0`				`src/pvx/cli/pvxtransient.py`
`--resample-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxtransient.py`
`--target-duration`	False				Target duration in seconds	`src/pvx/cli/pvxtransient.py`
`--time-stretch`	False	`1.0`				`src/pvx/cli/pvxtransient.py`
`--transient-threshold`	False	`1.6`				`src/pvx/cli/pvxtransient.py`

pvxunison.py

Flag	Required	Default	Choices	Action	Description	Source
`--detune-cents`	False	`14.0`			Total detune span in cents	`src/pvx/cli/pvxunison.py`
`--dry-mix`	False	`0.2`			Dry signal mix amount	`src/pvx/cli/pvxunison.py`
`--resample-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxunison.py`
`--voices`	False	`5`			Number of unison voices	`src/pvx/cli/pvxunison.py`
`--width`	False	`1.0`			Stereo width multiplier 0..2	`src/pvx/cli/pvxunison.py`

pvxvoc.py

Flag	Required	Default	Choices	Action	Description	S
`--ambient-phase-mix`	False	`0.5`			Random-phase blend when --phase-engine hybrid (0.0=propagated only, 1.0=random only; default: 0.5). Accepts scalar or control file (.csv/.json).	s
`--ambient-preset`	False			`store_true`	Convenience preset for ambient extreme stretch (random phase engine, onset-time-credit, transient preserve, conservative staging).	s
`--analysis-channel`	False	`mix`	`first, mix`		Channel strategy for F0 estimation with --target-f0 (default: mix)	s
`--auto-profile`	False			`store_true`	Analyze input and choose a profile automatically (speech/music/percussion/ambient/extreme).	s
`--auto-profile-lookahead-seconds`	False	`6.0`			Seconds of audio used when estimating --auto-profile (default: 6.0).	s
`--auto-segment-seconds`	False	`0.0`			Optional segment size in seconds for long jobs. When >0, processing runs per segment with crossfade assembly.	s
`--auto-transform`	False			`store_true`	Allow automatic transform selection when --transform is not explicitly set.	s

Flag	Required	Default	Choices	Action	Description	S
`--bit-depth`	False	`inherit`			Output bit-depth policy (default: inherit). Ignored when --subtype is set.	s
`--cents`	False				Pitch shift in cents (+1200 is one octave up). Accepts scalar or control file (.csv/.json).	s
`--checkpoint-dir`	False				Directory used to cache per-segment checkpoint chunks for resume workflows.	s
`--checkpoint-id`	False				Optional checkpoint run identifier (default: hash of input/settings).	s
`--clip`	False			`store_true`	Legacy alias: hard clip at +/-1.0 when set	s
`--coherence-strength`	False	`0.0`			Coherence lock strength in [0,1] (0=off, 1=full lock). Accepts scalar or control file (.csv/.json).	s
`--compander-attack-ms`	False	`8.0`			Compander attack time in ms	s
`--compander-compress-ratio`	False	`3.0`			Compander compression ratio (>=1)	s
`--compander-expand-ratio`	False	`1.8`			Compander expansion ratio (>=1)	s
`--compander-makeup-db`	False	`0.0`			Compander makeup gain in dB	s
`--compander-release-ms`	False	`120.0`			Compander release time in ms	s
`--compander-threshold-db`	False				Enable compander threshold in dBFS	s
`--compressor-attack-ms`	False	`10.0`			Compressor attack time in ms	s
`--compressor-makeup-db`	False	`0.0`			Compressor makeup gain in dB	s
`--compressor-ratio`	False	`4.0`			Compressor ratio (>=1)	s
`--compressor-release-ms`	False	`120.0`			Compressor release time in ms	s

Flag	Required	Default	Choices	Action	Description	S
`--compressor-threshold-db`	False				Enable compressor above threshold dBFS	s
`--cpu`	False			`store_true`	Alias for --device cpu.	s
`--cuda-device`	False	`0`			CUDA device index used when --device is auto/cuda (default: 0)	s
`--device`	False	`auto`	`auto, cpu, cuda`		Compute device: auto (prefer CUDA), cpu, or cuda	s
`--dither`	False	`none`			Dither policy before quantized writes (default: none)	s
`--dither-seed`	False				Deterministic RNG seed for dithering (default: random seed)	s
`--dry-run`	False			`store_true`	Resolve settings without writing files	s
`--example`	False				Print copy-paste example command(s) and exit.	s
`--expander-attack-ms`	False	`5.0`			Expander attack time in ms	s
`--expander-ratio`	False	`2.0`			Expander ratio (>=1)	s
`--expander-release-ms`	False	`120.0`			Expander release time in ms	s
`--expander-threshold-db`	False				Enable downward expander below threshold dBFS	s
`--explain-plan`	False			`store_true`	Print resolved processing plan JSON and exit without rendering audio.	s
`--extreme-stretch-threshold`	False	`2.0`			Auto-mode threshold for multistage activation (default: 2.0). Accepts scalar or control file (.csv/.json).	s
`--extreme-time-stretch`	False			`store_true`	Force multistage strategy even when ratio is moderate.	s
`--f0-max`	False	`1000.0`			Maximum F0 search bound in Hz (default: 1000)	s
`--f0-min`	False	`50.0`			Minimum F0 search bound in Hz (default: 50)	s
`--formant-lifter`	False	`32`			Cepstral lifter cutoff for formant envelope extraction (default: 32). Accepts scalar or control file (.csv/.json).	s

Flag	Required	Default	Choices	Action	Description	S
`--formant-max-gain-db`	False	`12.0`			Max per-bin formant correction gain in dB (default: 12). Accepts scalar or control file (.csv/.json).	s
`--formant-strength`	False	`1.0`			Formant correction blend 0..1 when pitch mode is formant-preserving (default: 1.0). Accepts scalar or control file (.csv/.json).	s
`--fourier-sync`	False			`store_true`	Enable fundamental frame locking. Uses generic short-time Fourier transforms with per-frame FFT sizes locked to detected F0.	s
`--fourier-sync-max-fft`	False	`8192`			Maximum frame FFT size for --fourier-sync (default: 8192). Accepts scalar or control file (.csv/.json).	s
`--fourier-sync-min-fft`	False	`256`			Minimum frame FFT size for --fourier-sync (default: 256). Accepts scalar or control file (.csv/.json).	s
`--fourier-sync-smooth`	False	`5`			Smoothing span (frames) for prescanned F0 track in --fourier-sync (default: 5). Accepts scalar or control file (.csv/.json).	s
`--gpu`	False			`store_true`	Alias for --device cuda.	s
`--guided`	False			`store_true`	Interactive guided mode for first-time users.	s
`--hard-clip-level`	False				Hard clip level in linear full-scale	s
`--hop-size`	False	`512`			Hop size in samples (default: 512). Accepts scalar or control file (.csv/.json).	s
`--interp`	False	`linear`			Interpolation mode for time-varying control signals loaded from CSV/JSON (default: linear).	s
`--kaiser-beta`	False	`14.0`			Kaiser window beta parameter used when --window kaiser (default: 14.0). Accepts scalar or control file (.csv/.json).	s
`--limiter-threshold`	False				Peak limiter threshold in linear full-scale	s
`--manifest-append`	False			`store_true`	Append entries to an existing --manifest-json file instead of replacing it.	s
`--manifest-json`	False				Write processing manifest JSON with per-file settings and outcomes.	s
`--max-stage-stretch`	False	`1.8`			Maximum per-stage ratio used in multistage mode (default: 1.8). Accepts scalar or control file (.csv/.json).	s
`--metadata-policy`	False	`none`			Output metadata policy: none, sidecar, or copy (sidecar implementation)	s
`--multires-ffts`	False	`1024,2048,4096`			Comma-separated FFT sizes for --multires-fusion (default: 1024,2048,4096)	s

Flag	Required	Default	Choices	Action	Description	S
`--multires-fusion`	False			`store_true`	Blend multiple FFT resolutions for each channel before pitch resampling.	s
`--multires-weights`	False				Comma-separated fusion weights for --multires-fusion (defaults to equal weights).	s
`--n-fft`	False	`2048`			FFT size (default: 2048). Accepts scalar or control file (.csv/.json).	s
`--no-center`	False			`store_true`	Disable center padding in STFT/ISTFT	s
`--no-onset-realign`	False			`store_true`	Disable fractional read-position realignment on onsets when --onset-time-credit is enabled.	s
`--no-progress`	False			`store_true`		s
`--normalize`	False	`none`	`none, peak, rms`		Output normalization mode	s
`--onset-credit-max`	False	`8.0`			Maximum accumulated onset time credit in analysis-frame units (default: 8.0). Accepts scalar or control file (.csv/.json).	s
`--onset-credit-pull`	False	`0.5`			Fraction of per-frame read advance removable while onset credit exists (0.0..1.0, default: 0.5). Accepts scalar or control file (.csv/.json).	s
`--onset-time-credit`	False			`store_true`	Enable onset-triggered time-credit scheduling to reduce transient smear during extreme stretching.	s
`--order`	False	`3`			Polynomial order for --interp polynomial (default: 3).	s
`--out`	False				Explicit output file path (single-input mode only). Alias: --out	s
`--output`	False				Explicit output file path (single-input mode only). Alias: --out	s
`--output-dir`	False				Directory for output files (default: same directory as each input)	s
`--output-format`	False				Output format/extension (e.g. wav, flac, aiff). Default: keep input extension.	s
`--overwrite`	False			`store_true`	Overwrite existing outputs	s
`--peak-dbfs`	False				Target peak dBFS when --normalize peak	s
`--phase-engine`	False	`propagate`			Phase synthesis engine: propagate (classic phase vocoder), hybrid (propagated + stochastic blend), random (ambient stochastic phase).	s
`--phase-locking`	False	`identity`	`off, identity`		Inter-bin phase locking mode for transient fidelity (default: identity)	s
`--phase-random-seed`	False				Optional deterministic seed for random/hybrid phase generation.	s
`--pitch`	False				Pitch shift in semitones (+12 is one octave up). Accepts scalar or control file (.csv/.json).	s

Flag	Required	Default	Choices	Action	Description	S
`--pitch-conf-min`	False	`0.0`			Minimum accepted map confidence (default: 0 disables gating).	s
`--pitch-follow-stdin`	False			`store_true`	Shortcut for --pitch-map-stdin (sidechain pitch-follow workflows).	s
`--pitch-lowconf-mode`	False	`hold`	`hold, unity, interp`		Low-confidence map handling mode (default: hold).	s
`--pitch-map`	False				CSV control map for time-varying stretch/pitch. Columns: start_sec,end_sec plus optional stretch,pitch_ratio/pitch_cents/pitch_semitones,confidence. Use '-' to read from stdin.	s
`--pitch-map-crossfade-ms`	False	`8.0`			Crossfade between processed map segments in milliseconds (default: 8.0).	s
`--pitch-map-smooth-ms`	False	`0.0`			Moving-average smoothing over map pitch ratios in milliseconds.	s
`--pitch-map-stdin`	False			`store_true`	Read control-map CSV from stdin.	s
`--pitch-mode`	False	`standard`	`standard, formant-preserving`		Pitch mode: standard shift or formant-preserving correction (default: standard)	s
`--pitch-shift-cents`	False				Pitch shift in cents (+1200 is one octave up). Accepts scalar or control file (.csv/.json).	s
`--pitch-shift-ratio`	False				Pitch ratio (>1 up, <1 down). Accepts decimals (1.5), integer ratios (3/2), expressions (2^(1/12)), or a control file (.csv/.json).	s
`--pitch-shift-semitones`	False				Pitch shift in semitones (+12 is one octave up). Accepts scalar or control file (.csv/.json).	s
`--preset`	False	`none`			High-level intent preset. Legacy: none/vocal/ambient/extreme. New: default/vocal_studio/drums_safe/extreme_ambient/stereo_coherent.	s
`--quality-profile`	False	`neutral`			Named tuning profile for vocoder defaults (default: neutral)	s
`--quiet`	False			`store_true`	Reduce output and hide status bars	s
`--ratio`	False				Pitch ratio (>1 up, <1 down). Accepts decimals (1.5), integer ratios (3/2), expressions (2^(1/12)), or a control file (.csv/.json).	s
`--ref-channel`	False	`0`			Reference channel index used by --stereo-mode ref_channel_lock (default: 0).	s
`--resample-mode`	False	`auto`	`auto, fft, linear`		Resampling engine (auto=fft if scipy available, else linear)	s
`--resume`	False			`store_true`	Reuse existing checkpoint chunks from --checkpoint-dir when available.	s

Flag	Required	Default	Choices	Action	Description	S
`--rms-dbfs`	False				Target RMS dBFS when --normalize rms	s
`--semitones`	False				Pitch shift in semitones (+12 is one octave up). Accepts scalar or control file (.csv/.json).	s
`--silent`	False			`store_true`	Suppress all console output	s
`--soft-clip-drive`	False	`1.0`			Soft clip drive amount (>0)	s
`--soft-clip-level`	False				Soft clip output ceiling in linear full-scale	s
`--soft-clip-type`	False	`tanh`	`tanh, arctan, cubic`		Soft clip transfer type	s
`--stdout`	False			`store_true`	Write processed audio to stdout stream (for piping); requires exactly one input	s
`--stereo-mode`	False	`independent`	`independent, mid_side_lock, ref_channel_lock`		Channel coherence strategy: independent (legacy), mid_side_lock (M/S-coupled), ref_channel_lock (phase-lock to reference channel).	s
`--stretch`	False				Alias for --time-stretch. Accepts scalar or control file (.csv/.json).	s
`--stretch-mode`	False	`auto`	`auto, standard, multistage`		Stretch strategy: standard (single pass), multistage (chained moderate passes), or auto (multistage only for extreme ratios; default: auto).	s
`--subtype`	False				Explicit libsndfile output subtype override (e.g., PCM_16, PCM_24, FLOAT)	s
`--suffix`	False	`_pv`			Suffix appended to output filename stem (default: _pv)	s
`--target-duration`	False				Absolute target duration in seconds (overrides --time-stretch)	s
`--target-f0`	False				Target fundamental frequency in Hz. Auto-estimates source F0 per file.	s
`--target-lufs`	False				Integrated loudness target in LUFS	s
`--target-pitch-shift-semitones`	False				Pitch shift in semitones (+12 is one octave up). Accepts scalar or control file (.csv/.json).	s
`--target-sample-rate`	False				Output sample rate in Hz (default: keep input rate)	s
`--time-stretch`	False	`1.0`			Final duration multiplier (1.0=unchanged, 2.0=2x longer). Accepts scalar or control file (.csv/.json).	s
`--time-stretch-factor`	False	`1.0`			Final duration multiplier (1.0=unchanged, 2.0=2x longer). Accepts scalar or control file (.csv/.json).	s

Flag	Required	Default	Choices	Action	Description	S
`--transform`	False	`fft`			Per-frame transform backend for STFT/ISTFT paths (default: fft; options: fft, dft, czt, dct, dst, hartley)	s
`--transient-crossfade-ms`	False	`10.0`			Crossfade duration for transient/steady stitching (default: 10 ms). Accepts scalar or control file (.csv/.json).	s
`--transient-mode`	False	`off`	`off, reset, hybrid, wsola`		Transient handling mode: off (none), reset (phase reset), hybrid (PV steady + WSOLA transients), or wsola (time-domain transient-safe path).	s
`--transient-preserve`	False			`store_true`	Enable transient phase resets based on spectral flux	s
`--transient-protect-ms`	False	`30.0`			Transient protection width in milliseconds (default: 30). Accepts scalar or control file (.csv/.json).	s
`--transient-sensitivity`	False	`0.5`			Transient detector sensitivity in [0,1] (higher catches more onsets). Accepts scalar or control file (.csv/.json).	s
`--transient-threshold`	False	`2.0`			Spectral-flux multiplier for transient detection (default: 2.0). Accepts scalar or control file (.csv/.json).	s
`--true-peak-max-dbtp`	False				Apply output gain trim to enforce max true-peak in dBTP	s
`--verbose`	False	`0`		`count`	Increase verbosity (repeat for extra detail)	s
`--verbosity`	False	`normal`			Console verbosity level	s
`--win-length`	False	`2048`			Window length in samples (default: 2048). Accepts scalar or control file (.csv/.json).	s
`--window`	False	`hann`			Window type (default: hann)	s

pvxwarp.py

Flag	Required	Default	Choices	Action	Description	Source
`--crossfade-ms`	False	`8.0`				`src/pvx/cli/pvxwarp.py`
`--map`	True				CSV map with start_sec,end_sec,stretch	`src/pvx/cli/pvxwarp.py`
`--resample-mode`	False	`auto`	`auto, fft, linear`			`src/pvx/cli/pvxwarp.py`

pvx Citation Quality

Source: citations.html

Citation quality report and BibTeX export.

`docs/references.bib`

Link-Type Summary

Link type	Count
`arxiv`	7
`doi`	33
`publisher_or_standard`	8
`scholar`	89
`web`	6

Scholar-Link Entries (upgrade candidates)

Year	Authors	Title	URL
1976	M. R. Portnoff	Implementation of the Digital Phase Vocoder Using the Fast Fourier Transform	link
1980	M. R. Portnoff	Time-Scale Modification of Speech Based on Short-Time Fourier Analysis	link
1986	R. J. McAulay; T. F. Quatieri	Speech Analysis/Synthesis Based on a Sinusoidal Representation	link
2002	C. Duxbury; M. Davies; M. Sandler	Improved Time-Scaling of Musical Audio Using Phase Locking at Transients	link
2000	J. Bonada	Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio	link
1985	N. Roucos; A. Wilgus	High Quality Time-Scale Modification for Speech	link
1993	W. Verhelst; M. Roelands	An Overlap-Add Technique Based on Waveform Similarity (WSOLA) for High Quality Time-Scale Modification of Speech	link
2002	S. Arfib; D. Keiler; U. Zölzer	DAFX: Digital Audio Effects (chapter references on pitch/time processing)	link
1967	A. M. Noll	Cepstrum Pitch Determination	link
1976	L. R. Rabiner; M. J. Cheng; A. E. Rosenberg; C. A. McGonegal	A Comparative Performance Study of Several Pitch Detection Algorithms	link
1995	D. Talkin	A Robust Algorithm for Pitch Tracking (RAPT)	link
1993	P. Boersma	Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-Noise Ratio of a Sampled Sound	link
2010	C. Schörkhuber; A. Klapuri	Constant-Q Transform Toolbox for Music Processing	link
2002	P. Flandrin; F. Auger; E. Chassande-Mottin	Time-Frequency Reassignment: From Principles to Algorithms	link
1991	S. Mann; S. Haykin	The Chirplet Transform: Physical Considerations	link
2007	T. Virtanen	Monaural Sound Source Separation by Nonnegative Matrix Factorization with Temporal Continuity and Sparseness Criteria	link
1949	N. Wiener	Extrapolation, Interpolation, and Smoothing of Stationary Time Series	link
1996	P. Scalart; J. V. Filho	Speech Enhancement Based on a Priori Signal to Noise Estimation	link
2012	Y. Yoshioka; T. Nakatani	Generalization of Multi-Channel Linear Prediction Methods for Blind MIMO Impulse Response Shortening	link
1997	S. Pulkki	Virtual Sound Source Positioning Using Vector Base Amplitude Panning	link

Year	Authors	Title	URL
2019	M. H. C. de Gesmundo et al.	ViSQOL v3: An Open Source Production Ready Objective Speech and Audio Metric	link
1989	R. Bristow-Johnson; M. Bogdanowicz	Phase Vocoder Done Right	link
1998	S. Disch	A New Phase Vocoder Technique for Time-Scale Modification of Audio Signals	link
1995	M. Puckette	Phase-Locked Vocoder	link
2012	B. Zavalishin	The Art of VA Filter Design (sections on phase and spectral processing)	link
2009	N. Moreau	Toolbox for Time-Scale Modification and Pitch-Shifting of Audio Signals	link
2013	M. Dolson	History of the Phase Vocoder	link
2010	D. S. Hamon; A. Lazarides	Improved WSOLA for Real-Time Time-Scale Modification	link
2015	M. Riess; A. R. Chhetri	Transient Preservation in Time-Scale Modification	link
2008	J. Bonada	Wide-Band Harmonic Sinusoidal Modeling for Time-Scale and Pitch-Scale Modification	link
2004	A. de Cheveigne	Pitch and Time Manipulation of Speech	link
2011	M. Le Roux; E. Vincent	Consistent Wiener Filtering for Audio Source Separation and Time-Frequency Processing	link
2003	J. Laroche	About this Phasiness Business	link
2014	J. Driedger; T. Pratzlich; M. Muller	Let It Bee - Towards NMF-Inspired Audio Mosaicing	link
2010	A. Roebel	A Shape-Invariant Phase Vocoder for Speech Transformation	link
2001	H. Kawahara; A. de Cheveigne	STRAIGHT, Exploitation of the Other Aspects of Vocal Source Information	link
2006	A. Klapuri	Multiple Fundamental Frequency Estimation by Harmonicity and Spectral Smoothness	link
2005	M. McLeod; G. Wyvill	A Smarter Way to Find Pitch	link
1999	S. Ahmadi; H. Spanias	Cepstrum-Based Pitch Detection Using FFT	link
2018	S. Bock; M. Korzeniowski	piano_transcription - F0 and onset tracking	link
2015	B. W. Schuller et al.	Paralinguistics and robust pitch tracking methods	link
2012	K. Dressler	Sinusoidal Extraction using Frequency-Domain Matching Pursuit	link
2000	S. Kum; C. L. Nikias	Robust F0 Estimation in Noise	link
2001	K. Grochenig	Foundations of Time-Frequency Analysis	link
1998	P. Flandrin	Time-Frequency/Time-Scale Analysis	link
1995	D. L. Donoho	De-Noising by Soft-Thresholding	link
1993	S. Mallat; Z. Zhang	Matching Pursuits with Time-Frequency Dictionaries	link
1989	L. Cohen	Time-Frequency Distributions - A Review	link
2010	A. V. Oppenheim; R. W. Schafer	Discrete-Time Signal Processing (STFT and filter-bank chapters)	link
2006	A. C. Gilbert; M. J. Strauss	Approximation of signals in sparse Fourier dictionaries	link
2019	M. Doring; M. M. Kokuer	Practical NSGT audio applications	link
2012	S. Essid; G. Richard	Musical signal analysis with reassignment methods	link
2007	P. Smaragdis	Convolutive Speech Bases and Their Application to Supervised Speech Separation	link
2009	C. Fevotte; N. Bertin; J.-L. Durrieu	Nonnegative Matrix Factorization with the Itakura-Saito Divergence	link
2000	A. Hyvarinen; E. Oja	Independent Component Analysis: Algorithms and Applications	link
2011	E. J. Candes; X. Li; Y. Ma; J. Wright	Robust Principal Component Analysis?	link
2011	N. Ono	Stable and fast update rules for independent low-rank matrix analysis based on auxiliary function technique	link
1995	A. J. Bell; T. J. Sejnowski	An Information-Maximization Approach to Blind Separation and Blind Deconvolution	link

Year	Authors	Title	URL
2019	J. Le Roux; S. Wisdom; H. Erdogan; J. R. Hershey	SDR half-baked or well done?	link
2019	A. Defossez et al.	Music Source Separation in the Waveform Domain	link
2018	N. Takahashi et al.	MMDenseLSTM for source separation	link
2019	F.-R. Stoter; S. Uhlich; A. Liutkus; Y. Mitsufuji	Open-Unmix	link
2001	I. Cohen; B. Berdugo	Speech Enhancement for Non-Stationary Noise Environments	link
1979	S. Boll	Suppression of Acoustic Noise in Speech Using Spectral Subtraction	link
2012	T. Yoshioka; T. Nakatani	Generalization of Multi-Channel Linear Prediction Methods for Blind MIMO Impulse Response Shortening	link
2012	N. D. Gaubitch; P. A. Naylor	Speech Dereverberation	link
2016	K. Kinoshita et al.	A summary of the REVERB challenge	link
2018	N. W. D. Evans et al.	The voicehome and CHiME challenges	link
2018	J.-M. Valin	A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement	link
2019	A. Pandey; D. Wang	A New Framework for CNN-Based Speech Enhancement in the Time Domain	link
2001	A. Rix et al.	Perceptual Evaluation of Speech Quality (PESQ)	link
2011	C. Taal et al.	An Algorithm for Intelligibility Prediction of Time-Frequency Weighted Noisy Speech	link
2012	A. Hines et al.	VISQOL: An Objective Speech Quality Model	link
2011	K. K. Paliwal	Importance of phase in speech enhancement	link
2013	P. C. Loizou	Speech Enhancement: Theory and Practice	link
2007	T. Lund	Loudness and True-Peak in Digital Audio	link
2020	S. Stables et al.	A Study of Loudness Normalization in Streaming Services	link
2018	M. Fenton	Practical Dynamics Processing for Modern Music Production	link
2011	J. Reiss	Under the Hood of a Dynamic Range Compressor	link
2012	D. Giannoulis; M. Massberg; J. Reiss	Digital Dynamic Range Compressor Design	link
2015	M. Ballou	Handbook for Sound Engineers (Loudness and Dynamics chapters)	link
2017	A. Gibiansky et al.	Deep Voice 2: Multi-Speaker Neural Text-to-Speech	link
2018	N. Kalchbrenner et al.	Efficient Neural Audio Synthesis	link
2019	R. Prenger et al.	WaveGlow: A Flow-based Generative Network for Speech Synthesis	link
2019	K. Kumar et al.	MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis	link
2020	J. Kong et al.	HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis	link
2017	Y. Wang et al.	Tacotron: Towards End-to-End Speech Synthesis	link
2021	S. Wisdom et al.	Differentiable Consistency Constraints for Improved Deep Speech Enhancement	link
2018	C. Donahue; J. McAuley; M. Puckette	Adversarial Audio Synthesis	link

pvx Group: Analysis, QA, and Automation

Source: groups/analysis_qa_and_automation.html

Theme: Analysis, QA, and Automation

Folder: `analysis_qa_and_automation` | **Algorithms:** 8

Subgroups: `core (8)`

Algorithm ID	Name	Subgroup	Module Path	Param keys
`analysis_qa_and_automation.onset_beat_downbeat_tracking`	Onset/beat/downbeat tracking	`core`	`src/pvx/algorithms/analysis_qa_and_automation/onset_beat_downbeat_tracking.py`	None (gener fault p
`analysis_qa_and_automation.key_chord_detection`	Key/chord detection	`core`	`src/pvx/algorithms/analysis_qa_and_automation/key_chord_detection.py`	None (gener fault p
`analysis_qa_and_automation.structure_segmentation_verse_chorus_sections`	Structure segmentation (verse/chorus/sections)	`core`	`src/pvx/algorithms/analysis_qa_and_automation/structure_segmentation_verse_chorus_sections.py`	None (gener fault p
`analysis_qa_and_automation.silence_speech_music_classifiers`	Silence/speech/music classifiers	`core`	`src/pvx/algorithms/analysis_qa_and_automation/silence_speech_music_classifiers.py`	None (gener fault p
`analysis_qa_and_automation.clip_hum_buzz_artifact_detection`	Clip/hum/buzz artifact detection	`core`	`src/pvx/algorithms/analysis_qa_and_automation/clip_hum_buzz_artifact_detection.py`	None (gener fault p
`analysis_qa_and_automation.pesq_stoi_visqol_quality_metrics`	PESQ/STOI/VISQOL quality metrics	`core`	`src/pvx/algorithms/analysis_qa_and_automation/pesq_stoi_visqol_quality_metrics.py`	None (gener fault p
`analysis_qa_and_automation.auto_parameter_tuning_bayesian_optimization`	Auto-parameter tuning (Bayesian optimization)	`core`	`src/pvx/algorithms/analysis_qa_and_automation/auto_parameter_tuning_bayesian_optimization.py`	`targ centr`
`analysis_qa_and_automation.batch_preset_recommendation_based_on_source_features`	Batch preset recommendation based on source features	`core`	`src/pvx/algorithms/analysis_qa_and_automation/batch_preset_recommendation_based_on_source_features.py`	None (gener fault p

pvx Group: Creative Spectral Effects

Source: groups/creative_spectral_effects.html

Theme: Creative Spectral Effects

Folder: `creative_spectral_effects` | **Algorithms:** 8

Subgroups: core (8)

Algorithm ID	Name	Subgroup	Module Path	Parameter keys
`creative_spectral_effects.cross_synthesis_vocoder`	Cross-synthesis vocoder	`core`	`src/pvx/algorithms/creative_spectral_effects/cross_synthesis_vocoder.py`	None (generic/default path)
`creative_spectral_effects.spectral_convolution_effects`	Spectral convolution effects	`core`	`src/pvx/algorithms/creative_spectral_effects/spectral_convolution_effects.py`	`kernel_size`
`creative_spectral_effects.spectral_freeze_banks`	Spectral freeze banks	`core`	`src/pvx/algorithms/creative_spectral_effects/spectral_freeze_banks.py`	`frame_ratio`
`creative_spectral_effects.spectral_blur_smear`	Spectral blur/smear	`core`	`src/pvx/algorithms/creative_spectral_effects/spectral_blur_smear.py`	None (generic/default path)
`creative_spectral_effects.phase_randomization_textures`	Phase randomization textures	`core`	`src/pvx/algorithms/creative_spectral_effects/phase_randomization_textures.py`	`strength`
`creative_spectral_effects.formant_painting_warping`	Formant painting/warping	`core`	`src/pvx/algorithms/creative_spectral_effects/formant_painting_warping.py`	`ratio`
`creative_spectral_effects.resonator_filterbank_morphing`	Resonator/filterbank morphing	`core`	`src/pvx/algorithms/creative_spectral_effects/resonator_filterbank_morphing.py`	None (generic/default path)
`creative_spectral_effects.spectral_contrast_exaggeration`	Spectral contrast exaggeration	`core`	`src/pvx/algorithms/creative_spectral_effects/spectral_contrast_exaggeration.py`	`amount`

pvx Group: Denoise and Restoration

Source: groups/denoise_and_restoration.html

Theme: Denoise and Restoration

Folder: `denoise_and_restoration` | **Algorithms:** 8

Subgroups: core (8)

Algorithm ID	Name	Subgroup	Module Path	Paramete
`denoise_and_restoration.wiener_denoising`	Wiener denoising	`core`	`src/pvx/algorithms/denoise_and_restoration/wiener_denoising.py`	None (ger fault path)
`denoise_and_restoration.mmse_stsa`	MMSE-STSA	`core`	`src/pvx/algorithms/denoise_and_restoration/mmse_stsa.py`	None (ger fault path)
`denoise_and_restoration.log_mmse`	Log-MMSE	`core`	`src/pvx/algorithms/denoise_and_restoration/log_mmse.py`	None (ger fault path)
`denoise_and_restoration.minimum_statistics_noise_tracking`	Minimum-statistics noise tracking	`core`	`src/pvx/algorithms/denoise_and_restoration/minimum_statistics_noise_tracking.py`	None (ger fault path)
`denoise_and_restoration.rnnoise_style_denoiser`	RNNoise-style denoiser	`core`	`src/pvx/algorithms/denoise_and_restoration/rnnoise_style_denoiser.py`	None (ger fault path)
`denoise_and_restoration.diffusion_based_speech_audio_denoise`	Diffusion-based speech/audio denoise	`core`	`src/pvx/algorithms/denoise_and_restoration/diffusion_based_speech_audio_denoise.py`	None (ger fault path)
`denoise_and_restoration.declip_via_sparse_reconstruction`	Declip via sparse reconstruction	`core`	`src/pvx/algorithms/denoise_and_restoration/declip_via_sparse_reconstruction.py`	`clip_th old`
`denoise_and_restoration.declick_decrackle_median_wavelet_interpolation`	Declick/decrackle (median/wavelet + interpolation)	`core`	`src/pvx/algorithms/denoise_and_restoration/declick_decrackle_median_wavelet_interpolation.py`	`spike_t old`

pvx Group: Dereverb and Room Correction

Source: groups/dereverb_and_room_correction.html

Theme: Dereverb and Room Correction

Folder: `dereverb_and_room_correction` | **Algorithms:** 8

Subgroups: `core (8)`

Algorithm ID	Name	Subgroup	Module Path	Parameter keys	Concept lin
`dereverb_and_room_correction.wpe_dereverberation`	WPE dereverberation	`core`	`src/pvx/algorithms/dereverb_and_room_correction/wpe_dereverberation.py`	`taps`, `delay`	Dereverbera
`dereverb_and_room_correction.spectral_decay_subtraction`	Spectral decay subtraction	`core`	`src/pvx/algorithms/dereverb_and_room_correction/spectral_decay_subtraction.py`	None (generic/default path)	None
`dereverb_and_room_correction.late_reverb_suppression_via_coherence`	Late reverb suppression via coherence	`core`	`src/pvx/algorithms/dereverb_and_room_correction/late_reverb_suppression_via_coherence.py`	None (generic/default path)	None
`dereverb_and_room_correction.room_impulse_inverse_filtering`	Room impulse inverse filtering	`core`	`src/pvx/algorithms/dereverb_and_room_correction/room_impulse_inverse_filtering.py`	None (generic/default path)	None
`dereverb_and_room_correction.multi_band_adaptive_deverb`	Multi-band adaptive deverb	`core`	`src/pvx/algorithms/dereverb_and_room_correction/multi_band_adaptive_deverb.py`	None (generic/default path)	None
`dereverb_and_room_correction.drr_guided_dereverb`	DRR-guided dereverb	`core`	`src/pvx/algorithms/dereverb_and_room_correction/drr_guided_dereverb.py`	None (generic/default path)	None
`dereverb_and_room_correction.blind_deconvolution_dereverb`	Blind deconvolution dereverb	`core`	`src/pvx/algorithms/dereverb_and_room_correction/blind_deconvolution_dereverb.py`	None (generic/default path)	Blind deconvoluti
`dereverb_and_room_correction.neural_dereverb_module`	Neural dereverb module	`core`	`src/pvx/algorithms/dereverb_and_room_correction/neural_dereverb_module.py`	None (generic/default path)	None

pvx Group: Dynamics and Loudness

Source: groups/dynamics_and_loudness.html

Theme: Dynamics and Loudness

Folder: `dynamics_and_loudness` | **Algorithms:** 8

Subgroups: `core (8)`

Algorithm ID	Name	Subgroup	Module Path	Parameter keys	Conc links
`dynamics_and_loudness.ebu_r128_normalization`	EBU R128 normalization	`core`	`src/pvx/algorithms/dynamics_and_loudness/ebu_r128_normalization.py`	`target_lufs`	EBU
`dynamics_and_loudness.itu_bs_1770_loudness_measurement_gating`	ITU BS.1770 loudness measurement/gating	`core`	`src/pvx/algorithms/dynamics_and_loudness/itu_bs_1770_loudness_measurement_gating.py`	`gate_lufs`	None
`dynamics_and_loudness.multi_band_compression`	Multi-band compression	`core`	`src/pvx/algorithms/dynamics_and_loudness/multi_band_compression.py`	None (generic/default path)	None
`dynamics_and_loudness.upward_compression`	Upward compression	`core`	`src/pvx/algorithms/dynamics_and_loudness/upward_compression.py`	`threshold_db`, `ratio`	None
`dynamics_and_loudness.transient_shaping`	Transient shaping	`core`	`src/pvx/algorithms/dynamics_and_loudness/transient_shaping.py`	`attack_boost`, `sustain`	None
`dynamics_and_loudness.spectral_dynamics_bin_wise_compressor_expander`	Spectral dynamics (bin-wise compressor/expander)	`core`	`src/pvx/algorithms/dynamics_and_loudness/spectral_dynamics_bin_wise_compressor_expander.py`	`threshold_db`, `ratio`	Com Expa
`dynamics_and_loudness.true_peak_limiting`	True-peak limiting	`core`	`src/pvx/algorithms/dynamics_and_loudness/true_peak_limiting.py`	`threshold`	True
`dynamics_and_loudness.lufs_target_mastering_chain`	LUFS-target mastering chain	`core`	`src/pvx/algorithms/dynamics_and_loudness/lufs_target_mastering_chain.py`	`target_lufs`	LUFS

pvx Group: Granular and Modulation

Source: groups/granular_and_modulation.html

Theme: Granular and Modulation

Folder: `granular_and_modulation` | **Algorithms:** 8

Subgroups: core (8)

Algorithm ID	Name	Subgroup	Module Path	Parameter keys	Co li
`granular_and_modulation.granular_time_stretch_engine`	Granular time-stretch engine	`core`	`src/pvx/algorithms/granular_and_modulation/granular_time_stretch_engine.py`	`stretch`, `grain`, `hop`	N
`granular_and_modulation.grain_cloud_pitch_textures`	Grain-cloud pitch textures	`core`	`src/pvx/algorithms/granular_and_modulation/grain_cloud_pitch_textures.py`	`seed`, `grain`, `count`, `stretch`	N
`granular_and_modulation.freeze_grain_morphing`	Freeze-grain morphing	`core`	`src/pvx/algorithms/granular_and_modulation/freeze_grain_morphing.py`	`grain`, `start`	N
`granular_and_modulation.am_fm_ring_modulation_blocks`	AM/FM/ring modulation blocks	`core`	`src/pvx/algorithms/granular_and_modulation/am_fm_ring_modulation_blocks.py`	`freq_hz`, `fm_depth`	Ri m tic
`granular_and_modulation.spectral_tremolo`	Spectral tremolo	`core`	`src/pvx/algorithms/granular_and_modulation/spectral_tremolo.py`	`lfo_hz`	N
`granular_and_modulation.formant_lfo_modulation`	Formant LFO modulation	`core`	`src/pvx/algorithms/granular_and_modulation/formant_lfo_modulation.py`	`lfo_hz`	Fo LF
`granular_and_modulation.rhythmic_gate_stutter_quantizer`	Rhythmic gate/stutter quantizer	`core`	`src/pvx/algorithms/granular_and_modulation/rhythmic_gate_stutter_quantizer.py`	`rate_hz`, `duty`	N
`granular_and_modulation.envelope_followed_modulation_routing`	Envelope-followed modulation routing	`core`	`src/pvx/algorithms/granular_and_modulation/envelope_followed_modulation_routing.py`	`depth`	N

pvx Group: Pitch Detection and Tracking

Source: groups/pitch_detection_and_tracking.html

Theme: Pitch Detection and Tracking

Folder: `pitch_detection_and_tracking` | **Algorithms:** 8

Subgroups: core (8)

Algorithm ID	Name	Subgroup	Module Path
`pitch_detection_and_tracking.yin`	YIN	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/yin.py`
`pitch_detection_and_tracking.pyin`	pYIN	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/pyin.py`
`pitch_detection_and_tracking.rapt`	RAPT	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/rapt.py`
`pitch_detection_and_tracking.swipe`	SWIPE	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/swipe.py`
`pitch_detection_and_tracking.harmonic_product_spectrum_hps`	Harmonic Product Spectrum (HPS)	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/harmonic_product_spe` `trum_hps.py`
`pitch_detection_and_tracking.subharmonic_summation`	Subharmonic summation	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/subharmonic_summatio`
`pitch_detection_and_tracking.crepe_style_neural_f0`	CREPE-style neural F0	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/crepe_style_neural_f`
`pitch_detection_and_tracking.viterbi_smoothed_pitch_contour_tracking`	Viterbi-smoothed pitch contour tracking	`core`	`src/pvx/algorithms/pitch_detection_and_tracking/viterbi_smoothed_pit` `contour_tracking.py`

pvx Group: Retune and Intonation

Source: groups/retune_and_intonation.html

Theme: Retune and Intonation

Folder: `retune_and_intonation` | **Algorithms:** 8

Subgroups: `core (8)`

Algorithm ID	Name	Subgroup	Module Path	Parameter keys
`retune_and_intonation.chord_aware_retuning`	Chord-aware retuning	`core`	`src/pvx/algorithms/retune_and_intonation/chord_aware_retuning.py`	`strength`
`retune_and_intonation.key_aware_retuning_with_confidence_weighting`	Key-aware retuning with confidence weighting	`core`	`src/pvx/algorithms/retune_and_intonation/key_aware_retuning_with_confidence_weighting.py`	None (generic/default path)
`retune_and_intonation.just_intonation_mapping_per_key_center`	Just intonation mapping per key center	`core`	`src/pvx/algorithms/retune_and_intonation/just_intonation_mapping_per_key_center.py`	None (generic/default path)
`retune_and_intonation.adaptive_intonation_context_sensitive_intervals`	Adaptive intonation (context-sensitive intervals)	`core`	`src/pvx/algorithms/retune_and_intonation/adaptive_intonation_context_sensitive_intervals.py`	None (generic/default path)
`retune_and_intonation.scala_mts_scale_import_and_quantization`	Scala/MTS scale import and quantization	`core`	`src/pvx/algorithms/retune_and_intonation/scala_mts_scale_import_and_quantization.py`	None (generic/default path)
`retune_and_intonation.time_varying_cents_maps`	Time-varying cents maps	`core`	`src/pvx/algorithms/retune_and_intonation/time_varying_cents_maps.py`	`cents_curve`
`retune_and_intonation.vibrato_preserving_correction`	Vibrato-preserving correction	`core`	`src/pvx/algorithms/retune_and_intonation/vibrato_preserving_correction.py`	`strength`
`retune_and_intonation.portamento_aware_retune_curves`	Portamento-aware retune curves	`core`	`src/pvx/algorithms/retune_and_intonation/portamento_aware_retune_curves.py`	`max_semitone_step`, `compression`

pvx Group: Separation and Decomposition

Source: groups/separation_and_decomposition.html

Theme: Separation and Decomposition

Folder: `separation_and_decomposition` | **Algorithms:** 8

Subgroups: `core (8)`

Algorithm ID	Name	Subgroup	Module Path
`separation_and_decomposition.rpca_hpss`	RPCA HPSS	`core`	`src/pvx/algorithms/separation_and_decomposition/rpca_hpss.py`
`separation_and_decomposition.nmf_decomposition`	NMF decomposition	`core`	`src/pvx/algorithms/separation_and_decomposition/nmf_decomposition.py`
`separation_and_decomposition.ica_bss_for_multichannel_stems`	ICA/BSS for multichannel stems	`core`	`src/pvx/algorithms/separation_and_decomposition/ica_bss_for_multichannel_stems.py`
`separation_and_decomposition.sinusoidal_residual_transient_decomposition`	Sinusoidal+residual+transient decomposition	`core`	`src/pvx/algorithms/separation_and_decomposition/sinusoidal_residual_transient_decomposition.py`
`separation_and_decomposition.demucs_style_stem_separation_backend`	Demucs-style stem separation backend	`core`	`src/pvx/algorithms/separation_and_decomposition/demucs_style_stem_separation_backend.py`
`separation_and_decomposition.u_net_vocal_accompaniment_split`	U-Net vocal/accompaniment split	`core`	`src/pvx/algorithms/separation_and_decomposition/u_net_vocal_accompaniment_split.py`
`separation_and_decomposition.tensor_decomposition_cp_tucker`	Tensor decomposition (CP/Tucker)	`core`	`src/pvx/algorithms/separation_and_decomposition/tensor_decomposition_cp_tucker.py`
`separation_and_decomposition.probabilistic_latent_component_separation`	Probabilistic latent component separation	`core`	`src/pvx/algorithms/separation_and_decomposition/probabilistic_latent_component_separation.py`

pvx Group: Spatial and Multichannel

Source: groups/spatial_and_multichannel.html

Theme: Spatial and Multichannel

Folder: `spatial_and_multichannel` | **Algorithms:** 24

Subgroups: `imaging_and_panning` (6) `phase_vocoder_spatial` (6) `multichannel_restoration` (6) `creative_spatial_fx` (6)

Algorithm ID	Name	Subgroup	Module Path
`spatial_and_multichannel.vbap_adaptive_panning`	VBAP adaptive panning	`imaging_and_panning`	`src/pvx/algorithms/spatial_and_multichannel/imaging_ning.py`
`spatial_and_multichannel.dbap_distance_based_amplitude_panning`	DBAP (distance-based amplitude panning)	`imaging_and_panning`	`src/pvx/algorithms/spatial_and_multichannel/imaging_plitude_panning.py`
`spatial_and_multichannel.binaural_itd_ild_synthesis`	Binaural ITD/ILD synthesis	`imaging_and_panning`	`src/pvx/algorithms/spatial_and_multichannel/imaging_sis.py`
`spatial_and_multichannel.transaural_crosstalk_cancellation`	Transaural crosstalk cancellation	`imaging_and_panning`	`src/pvx/algorithms/spatial_and_multichannel/imaging_cancellation.py`
`spatial_and_multichannel.stereo_width_frequency_dependent_control`	Stereo width (frequency-dependent control)	`imaging_and_panning`	`src/pvx/algorithms/spatial_and_multichannel/imaging_cy_dependent_control.py`
`spatial_and_multichannel.phase_aligned_mid_side_field_rotation`	Phase-aligned mid/side field rotation	`imaging_and_panning`	`src/pvx/algorithms/spatial_and_multichannel/imaging_field_rotation.py`
`spatial_and_multichannel.pvx_interchannel_phase_locking`	pvx inter-channel phase locking	`phase_vocoder_spatial`	`src/pvx/algorithms/spatial_and_multichannel/phase_vc nel_phase_locking.py`
`spatial_and_multichannel.pvx_spatial_transient_preservation`	pvx spatial transient preservation	`phase_vocoder_spatial`	`src/pvx/algorithms/spatial_and_multichannel/phase_vc t_preservation.py`
`spatial_and_multichannel.pvx_interaural_coherence_shaping`	pvx interaural coherence shaping	`phase_vocoder_spatial`	`src/pvx/algorithms/spatial_and_multichannel/phase_vc ence_shaping.py`
`spatial_and_multichannel.pvx_directional_spectral_warp`	pvx directional spectral warp	`phase_vocoder_spatial`	`src/pvx/algorithms/spatial_and_multichannel/phase_vc tral_warp.py`
`spatial_and_multichannel.pvx_multichannel_time_alignment`	pvx multi-channel time alignment	`phase_vocoder_spatial`	`src/pvx/algorithms/spatial_and_multichannel/phase_vc time_alignment.py`
`spatial_and_multichannel.pvx_spatial_freeze_and_trajectory`	pvx spatial freeze and trajectory	`phase_vocoder_spatial`	`src/pvx/algorithms/spatial_and_multichannel/phase_vc tial_freeze_and_trajectory.py`

Algorithm ID	Name	Subgroup	Module Path
`spatial_and_multichannel.multichannel_wiener_postfilter`	Multichannel Wiener postfilter	`multichannel_restoration`	`src/pvx/algorithms/spatial_and_multichannel/multicha nel_wiener_postfilter.py`
`spatial_and_multichannel.coherence_based_dereverb_multichannel`	Coherence-based dereverb (multichannel)	`multichannel_restoration`	`src/pvx/algorithms/spatial_and_multichannel/multicha ence_based_dereverb_multichannel.py`
`spatial_and_multichannel.multichannel_noise_psd_tracking`	Multichannel noise PSD tracking	`multichannel_restoration`	`src/pvx/algorithms/spatial_and_multichannel/multicha nel_noise_psd_tracking.py`
`spatial_and_multichannel.phase_consistent_multichannel_denoise`	Phase-consistent multichannel denoise	`multichannel_restoration`	`src/pvx/algorithms/spatial_and_multichannel/multicha multichannel_denoise.py`
`spatial_and_multichannel.microphone_array_calibration_tones`	Microphone-array calibration tones	`multichannel_restoration`	`src/pvx/algorithms/spatial_and_multichannel/multicha calibration_tones.py`
`spatial_and_multichannel.cross_channel_click_pop_repair`	Cross-channel click/pop repair	`multichannel_restoration`	`src/pvx/algorithms/spatial_and_multichannel/multicha nel_click_pop_repair.py`
`spatial_and_multichannel.rotating_speaker_doppler_field`	Rotating-speaker Doppler field	`creative_spatial_fx`	`src/pvx/algorithms/spatial_and_multichannel/creative doppler_field.py`
`spatial_and_multichannel.binaural_motion_trajectory_designer`	Binaural motion trajectory designer	`creative_spatial_fx`	`src/pvx/algorithms/spatial_and_multichannel/creative tory_designer.py`
`spatial_and_multichannel.stochastic_spatial_diffusion_cloud`	Stochastic spatial diffusion cloud	`creative_spatial_fx`	`src/pvx/algorithms/spatial_and_multichannel/creative fusion_cloud.py`
`spatial_and_multichannel.decorrelated_reverb_upmix`	Decorrelated reverb upmix	`creative_spatial_fx`	`src/pvx/algorithms/spatial_and_multichannel/creative mix.py`
`spatial_and_multichannel.spectral_spatial_granulator`	Spectral spatial granulator	`creative_spatial_fx`	`src/pvx/algorithms/spatial_and_multichannel/creative lator.py`
`spatial_and_multichannel.spatial_freeze_resynthesis`	Spatial freeze resynthesis	`creative_spatial_fx`	`src/pvx/algorithms/spatial_and_multichannel/creative sis.py`

pvx Group: Spectral and Time-Frequency Transforms

Source: groups/spectral_time_frequency_transforms.html

Theme: Spectral and Time-Frequency Transforms

Folder: `spectral_time_frequency_transforms` | **Algorithms:** 8

Subgroups: core (8)

Algorithm ID	Name	Subgroup	Module Path	Pa ke
`spectral_time_frequency_transforms.constant_q_transform_cqt_processing`	Constant-Q Transform (CQT) processing	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/constant_q_transform_cqt_processing.py`	N (g fa
`spectral_time_frequency_transforms.variable_q_transform_vqt`	Variable-Q Transform (VQT)	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/variable_q_transform_vqt.py`	N (g fa
`spectral_time_frequency_transforms.nsgt_based_processing`	NSGT-based processing	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/nsgt_based_processing.py`	N (g fa
`spectral_time_frequency_transforms.reassigned_spectrogram_methods`	Reassigned spectrogram methods	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/reassigned_spectrogram_methods.py`	N (g fa
`spectral_time_frequency_transforms.synchrosqueezed_stft`	Synchrosqueezed STFT	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/synchrosqueezed_stft.py`	N (g fa
`spectral_time_frequency_transforms.chirplet_transform_analysis`	Chirplet transform analysis	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/chirplet_transform_analysis.py`	N (g fa
`spectral_time_frequency_transforms.wavelet_packet_processing`	Wavelet packet processing	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/wavelet_packet_processing.py`	N (g fa
`spectral_time_frequency_transforms.multi_window_stft_fusion`	Multi-window STFT fusion	`core`	`src/pvx/algorithms/spectral_time_frequency_transforms/multi_window_stft_fusion.py`	N (g fa

pvx Group: Time-Scale and Pitch Core

Source: groups/time_scale_and_pitch_core.html

Theme: Time-Scale and Pitch Core

Folder: `time_scale_and_pitch_core` | **Algorithms:** 7

Subgroups: `core (7)`

Algorithm ID	Name	Subgroup	Module Path	Parameter keys
`time_scale_and_pitch_core.wsola_waveform_similarity_overlap_add`	WSOLA (Waveform Similarity Overlap-Add)	`core`	`src/pvx/algorithms/time_scale_and_pitch_core/wsola_waveform_similarity_overlap_add.py`	`stretch`, `grain_size`, `hop`
`time_scale_and_pitch_core.td_psola`	TD-PSOLA	`core`	`src/pvx/algorithms/time_scale_and_pitch_core/td_psola.py`	`semitones`, `stretch`
`time_scale_and_pitch_core.lp_psola`	LP-PSOLA	`core`	`src/pvx/algorithms/time_scale_and_pitch_core/lp_psola.py`	`semitones`
`time_scale_and_pitch_core.multi_resolution_phase_vocoder`	Multi-resolution phase vocoder	`core`	`src/pvx/algorithms/time_scale_and_pitch_core/multi_resolution_phase_vocoder.py`	`stretch`
`time_scale_and_pitch_core.harmonic_percussive_split_tsm`	Harmonic/percussive split TSM	`core`	`src/pvx/algorithms/time_scale_and_pitch_core/harmonic_percussive_split_tsm.py`	`harmonic_stretch`, `percussive_stretch`
`time_scale_and_pitch_core.beat_synchronous_time_warping`	Beat-synchronous time warping	`core`	`src/pvx/algorithms/time_scale_and_pitch_core/beat_synchronous_time_warping.py`	`stretch`
`time_scale_and_pitch_core.nonlinear_time_maps`	Nonlinear time maps (curves, anchors, spline timing)	`core`	`src/pvx/algorithms/time_scale_and_pitch_core/nonlinear_time_maps.py`	`curve`, `stretch`, `fmin`, `fmax`