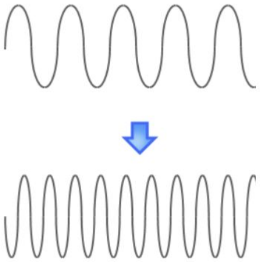
Evolving Constants by Rewriting Source Code

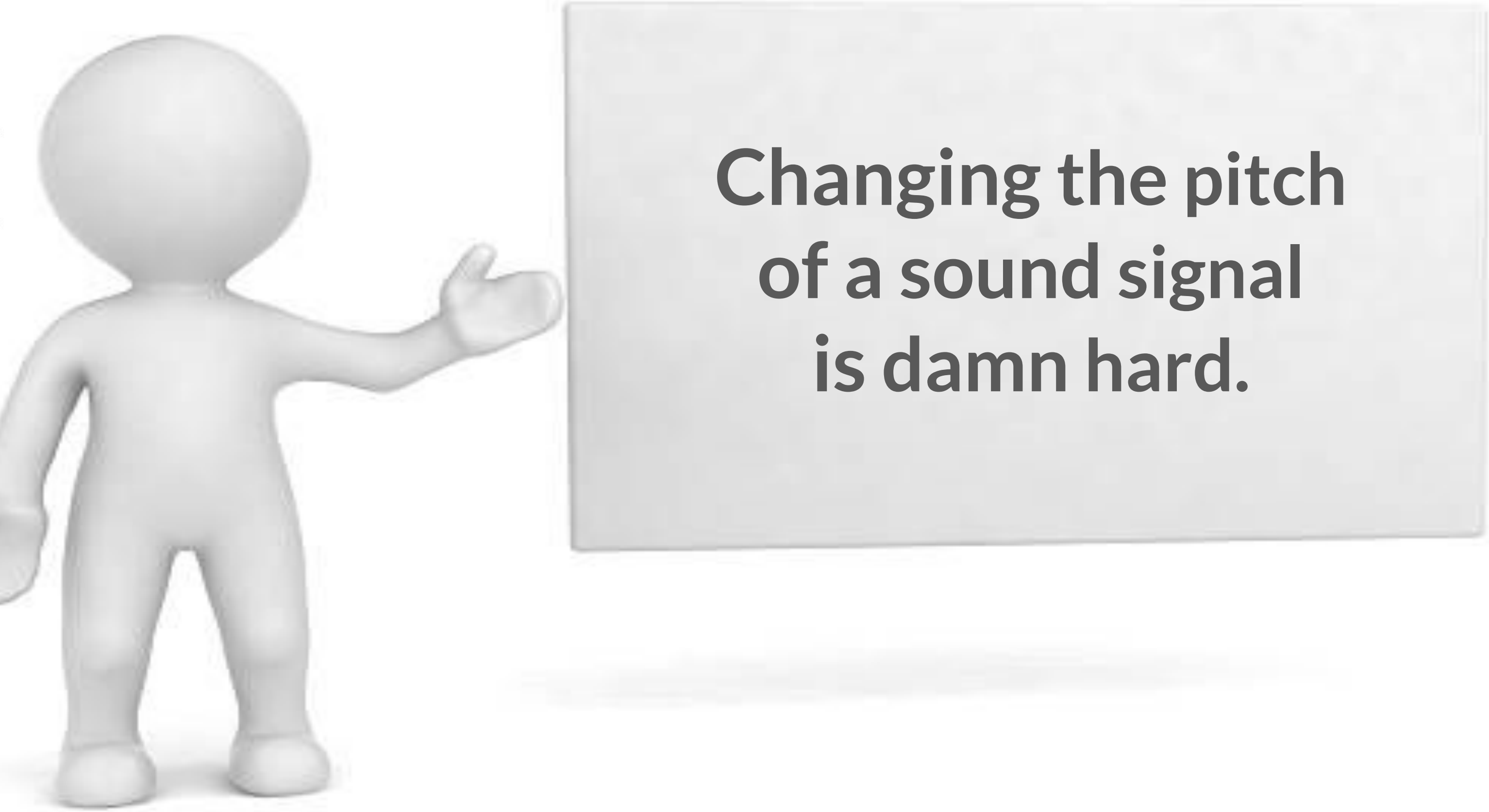
Dconf 2025 London — Guillaume Piolat

Nature

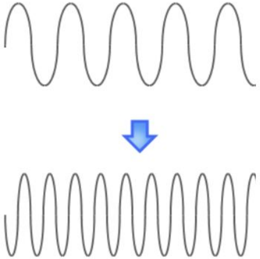
doesn’t have

low-latency pitch-shifting.





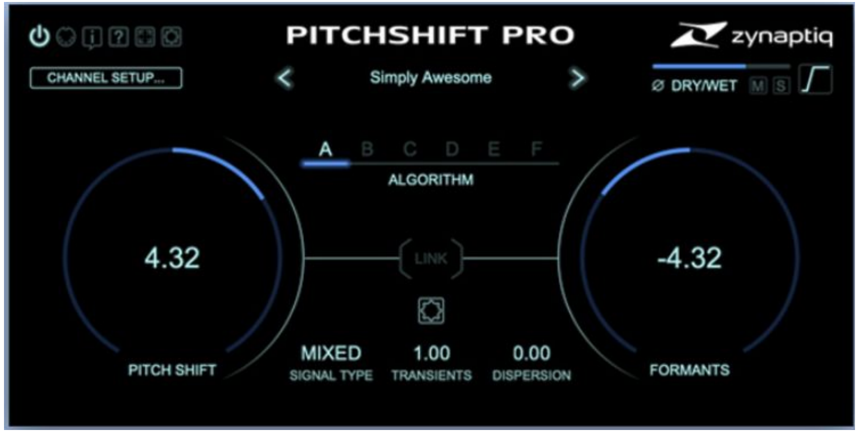
Changing the pitch
of a sound signal
is damn hard.



Changing the pitch of a sound signal is damn hard.

Especially real-time, with polyphonic input.

Zynaptiq Pitchshift Pro (2024)



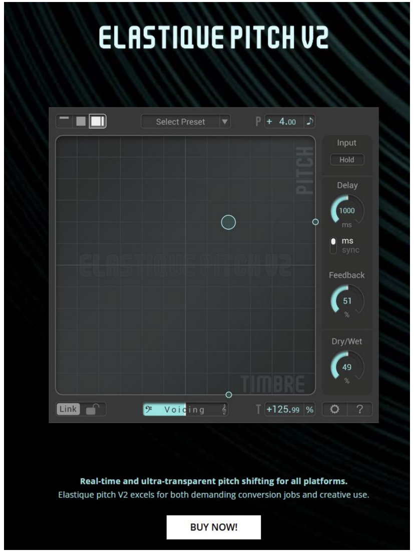
PITCH SHIFT 4.32

FORMANTS -4.32

SIGNAL TYPE MIXED

TRANSIENTS 1.00

DISPERSION 0.00



P + 4.00

Delay 1000 ms

ms sync

Feedback 51 %

Dry/Wet 49 %

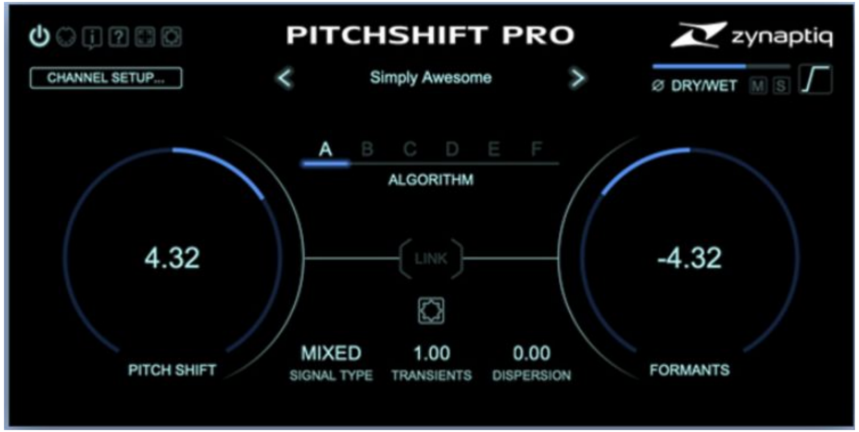
T +125.99 %

Real-time and ultra-transparent pitch shifting for all platforms.
Elastique pitch V2 excels for both demanding conversion jobs and creative use.

Zplane Elastique Pro V3 Engine (2015)

Zynaptiq Pitchshift Pro (2024)

🇩🇪



PITCHSHIFT PRO

zynaptiq

CHANNEL SETUP...

Simply Awesome

Ø DRY/WET M S

ALGORITHM: A B C D E F

PITCH SHIFT: 4.32

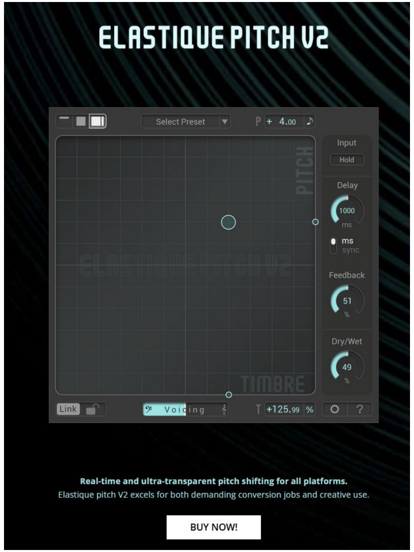
LINK

FORMANTS: -4.32

SIGNAL TYPE: MIXED

TRANSIENTS: 1.00

DISPERSION: 0.00



ELASTIQUE PITCH V2

Select Preset

P + 4.00

Input: Hold

Delay: 1000 ms

ms / sync

Feedback: 51 %

Dry/Wet: 49 %

T +125.99 %

Real-time and ultra-transparent pitch shifting for all platforms.

Elastique pitch V2 excels for both demanding conversion jobs and creative use.

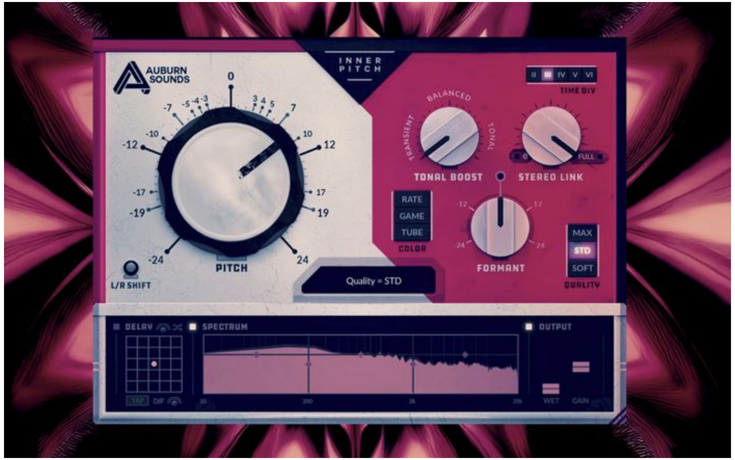
BUY NOW!

Zplane Elastique Pro V3 Engine (2015)

🇩🇪

Inner Pitch v1 (2023)

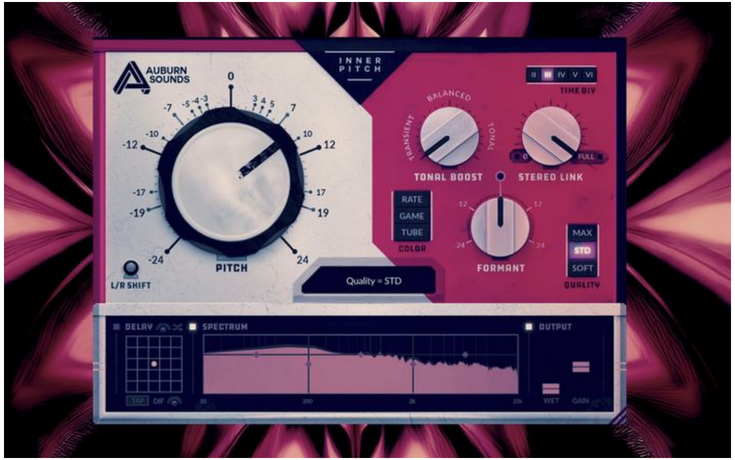
- 17ms latency



Quality = STD

Inner Pitch v1 (2023)

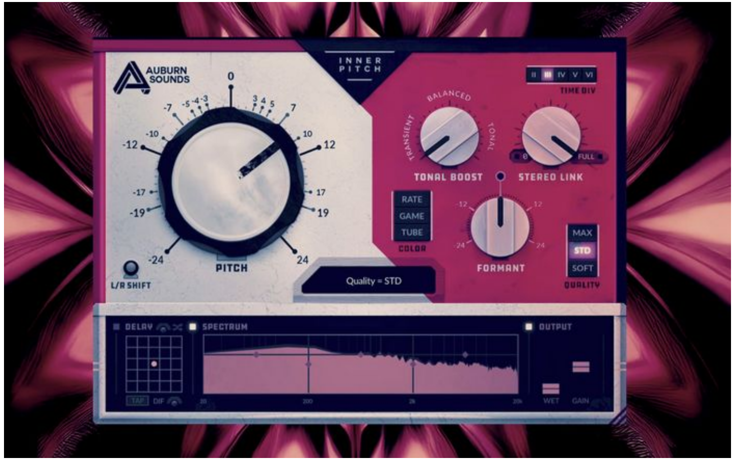
- 17ms latency
- Competitive and written in D



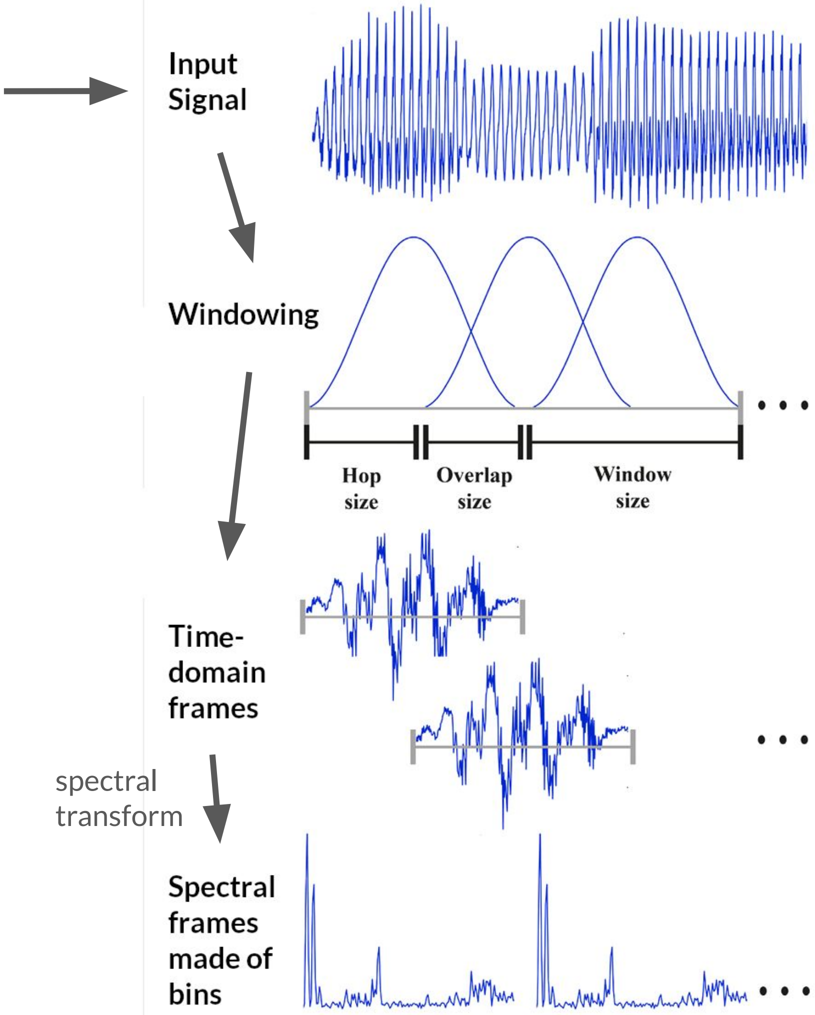
Quality = STD

Inner Pitch v1 (2023)

- 17ms latency
- Competitive and written in D
- All the State of the Art algorithms are spectral aka **Phase Vocoders**



What does a Phase Vocoder do anyway?



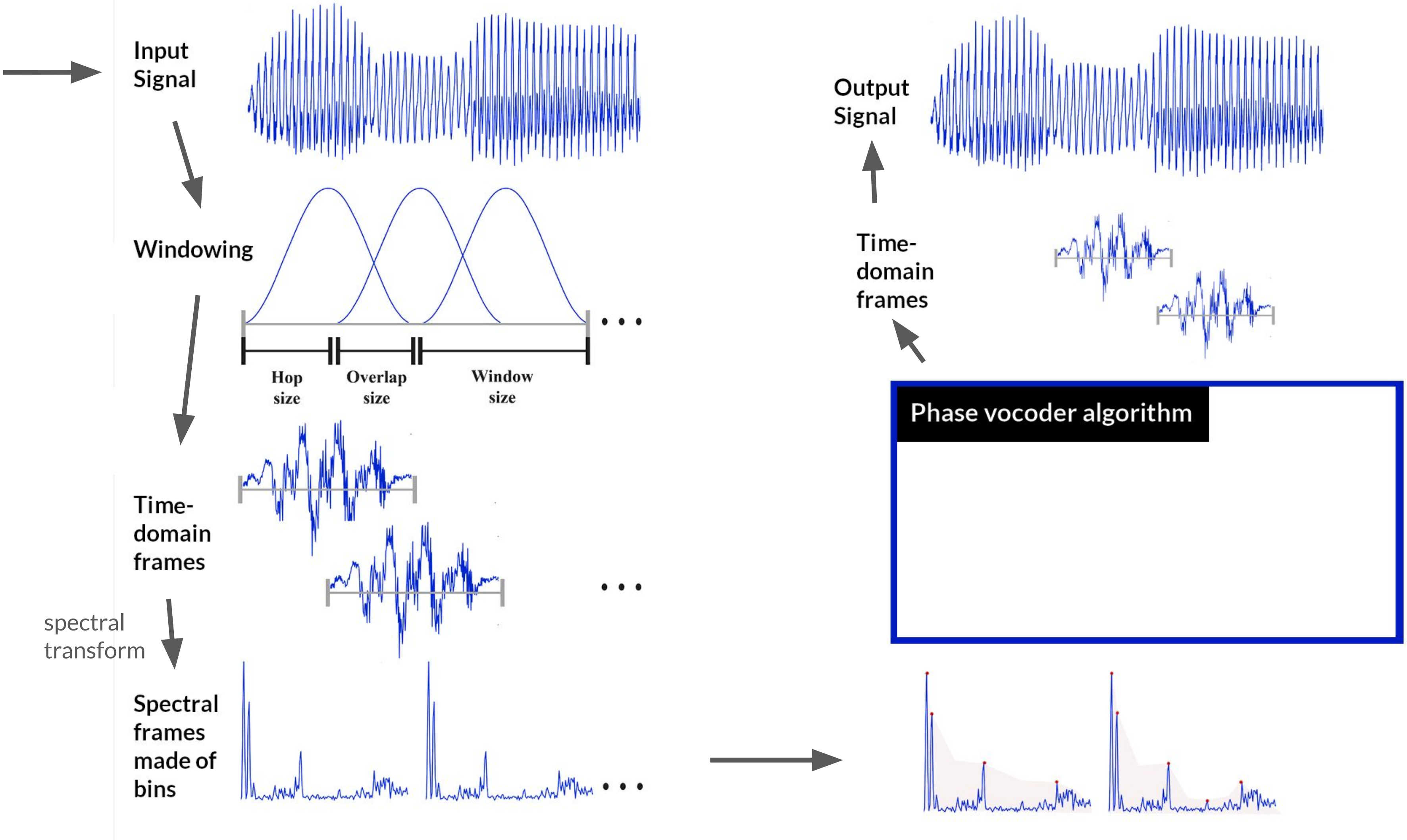
Input Signal

Windowing

Time-domain frames

spectral transform

Spectral frames made of bins



Input Signal

Windowing

Time-domain frames

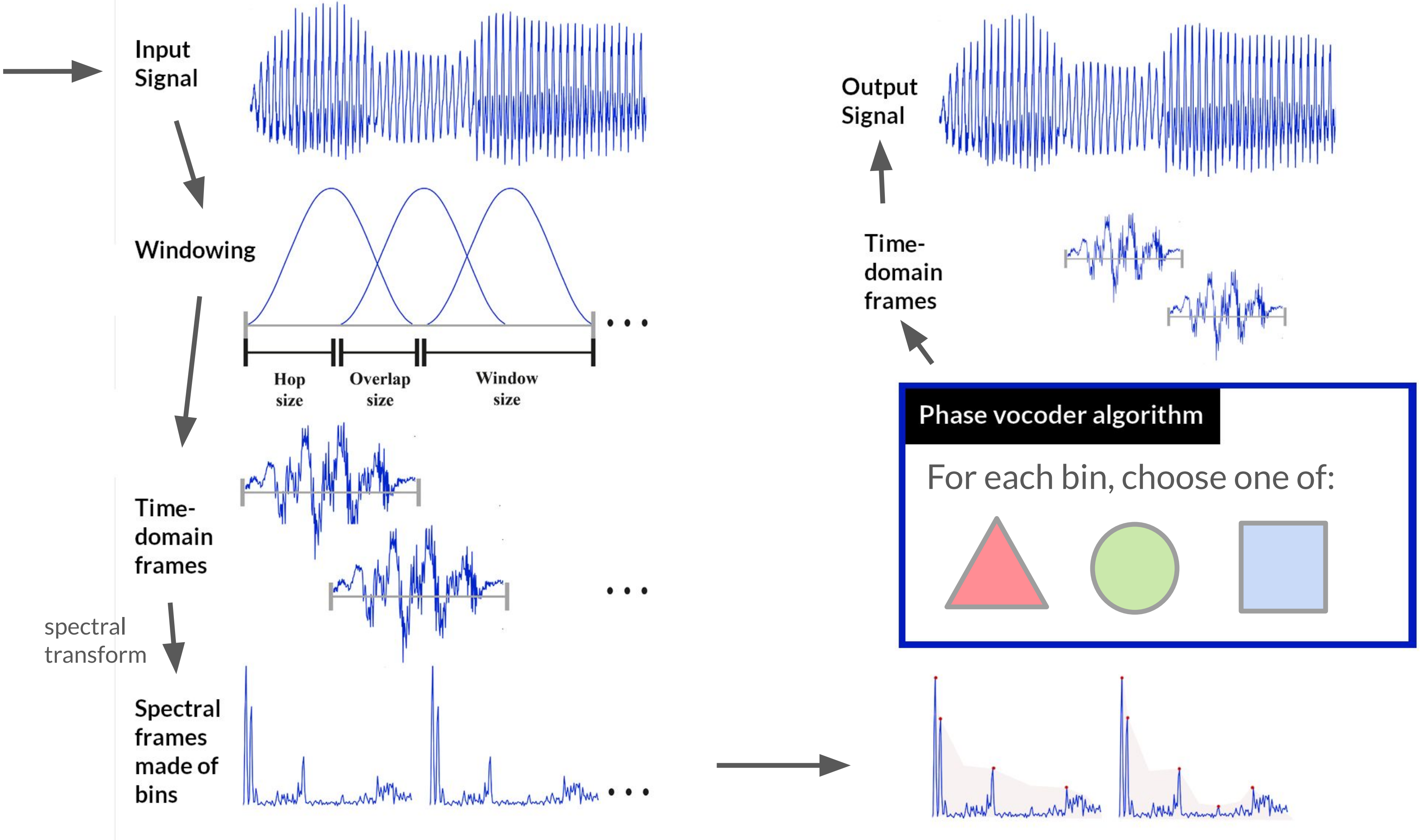
spectral transform

Spectral frames made of bins

Phase vocoder algorithm

Time-domain frames

Output Signal



Input Signal

Windowing

Time-domain frames

spectral transform

Spectral frames made of bins

Phase vocoder algorithm

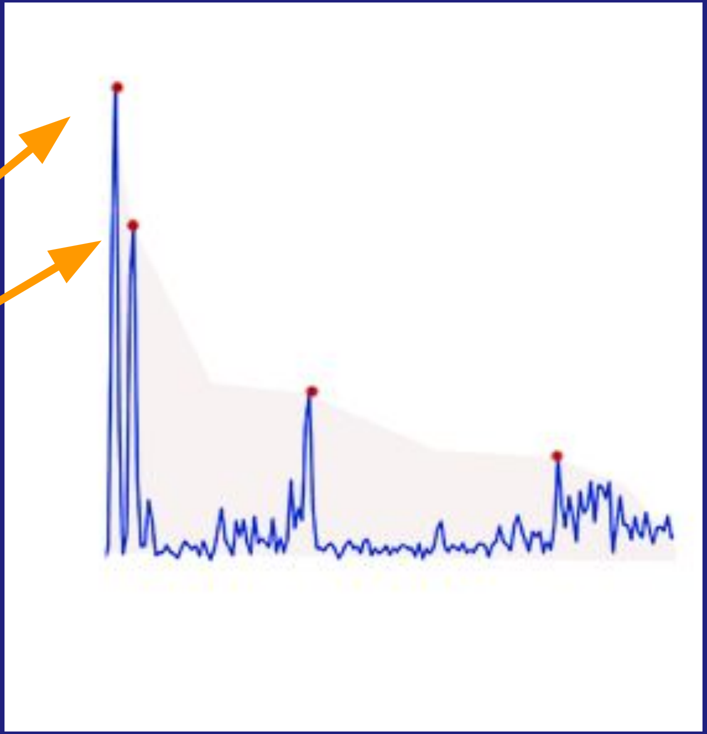
For each bin, choose one of:

△ ○ □

Time-domain frames

Output Signal

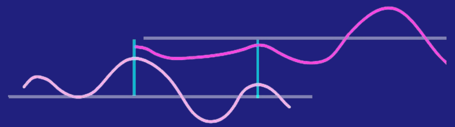
Phase propagation in the Phase Vocoder



Foreach bin in the spectral frame

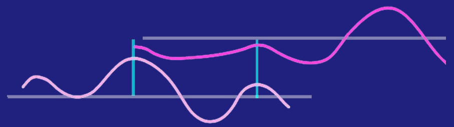
Phase propagation in the Phase Vocoder

- HORIZONTAL CHOICE
  Favors continuity with previous frame.

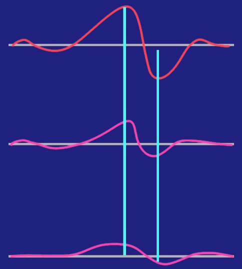


Phase propagation in the Phase Vocoder

● HORIZONTAL CHOICE
Favors continuity with previous frame.

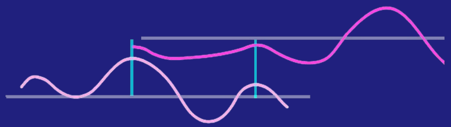


■ VERTICAL CHOICE
Do like strong neighbour bins do.

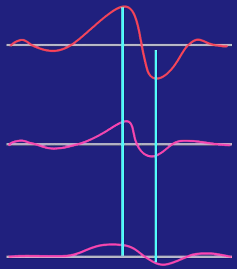


Phase propagation in the Phase Vocoder

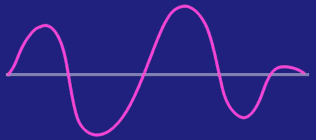
● HORIZONTAL CHOICE
Favors continuity with previous frame.



■ VERTICAL CHOICE
Do like strong neighbour bins do.

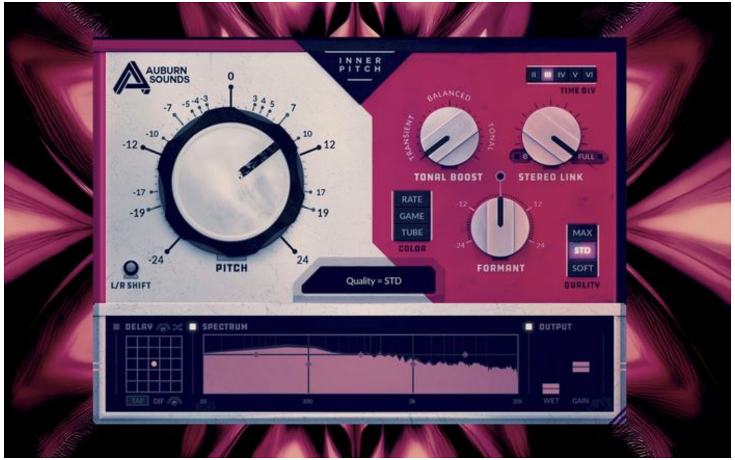


▲ TRANSIENT CHOICE
Favors this bin phase information.
Ignore neighbours or previous frame.



End results in 2023

- Complex algorithm

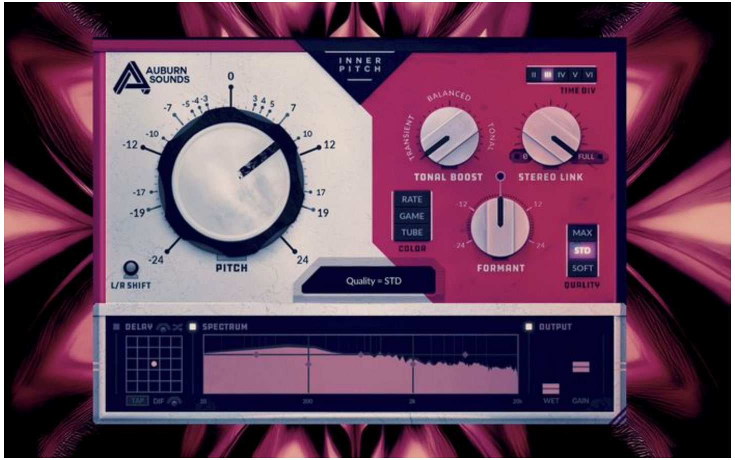


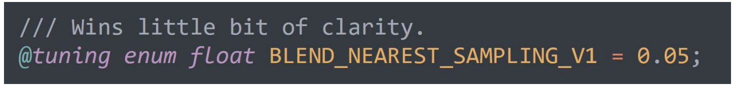
Quality = STD

End results in 2023

- Complex algorithm

- ~100 magic constants this time





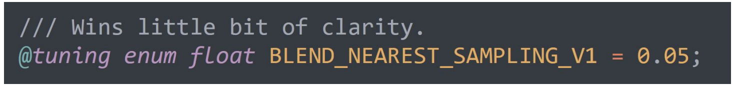
```
/// Wins little bit of clarity.
@tuning enum float BLEND_NEAREST_SAMPLING_V1 = 0.05;
```

Such as this one.

End results in 2023

- Complex algorithm
- ~100 magic constants this time
- Most of the time is spent **making these constants appear**, and **finding good values** for them.

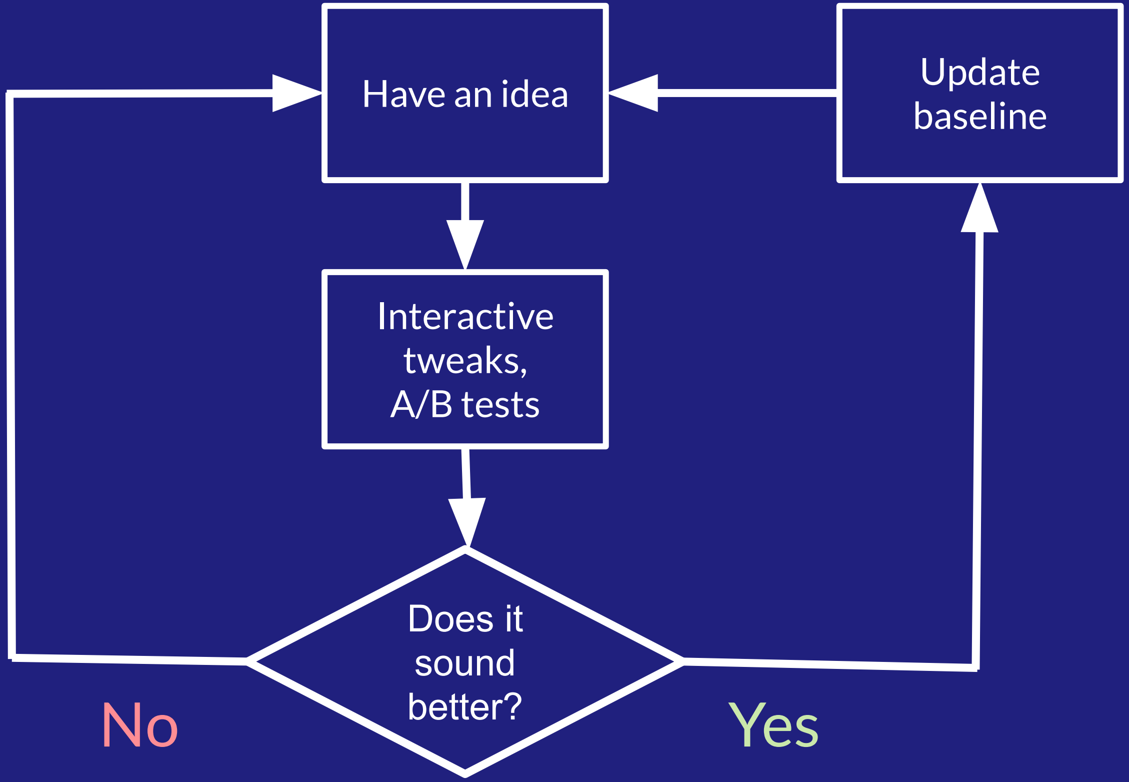
Quality = STD



```
/// Wins little bit of clarity.
@tuning enum float BLEND_NEAREST_SAMPLING_V1 = 0.05;
```

Such as this one.

Tuning process



A sort of manual gradient descent

- Need to retune already tuned constants.

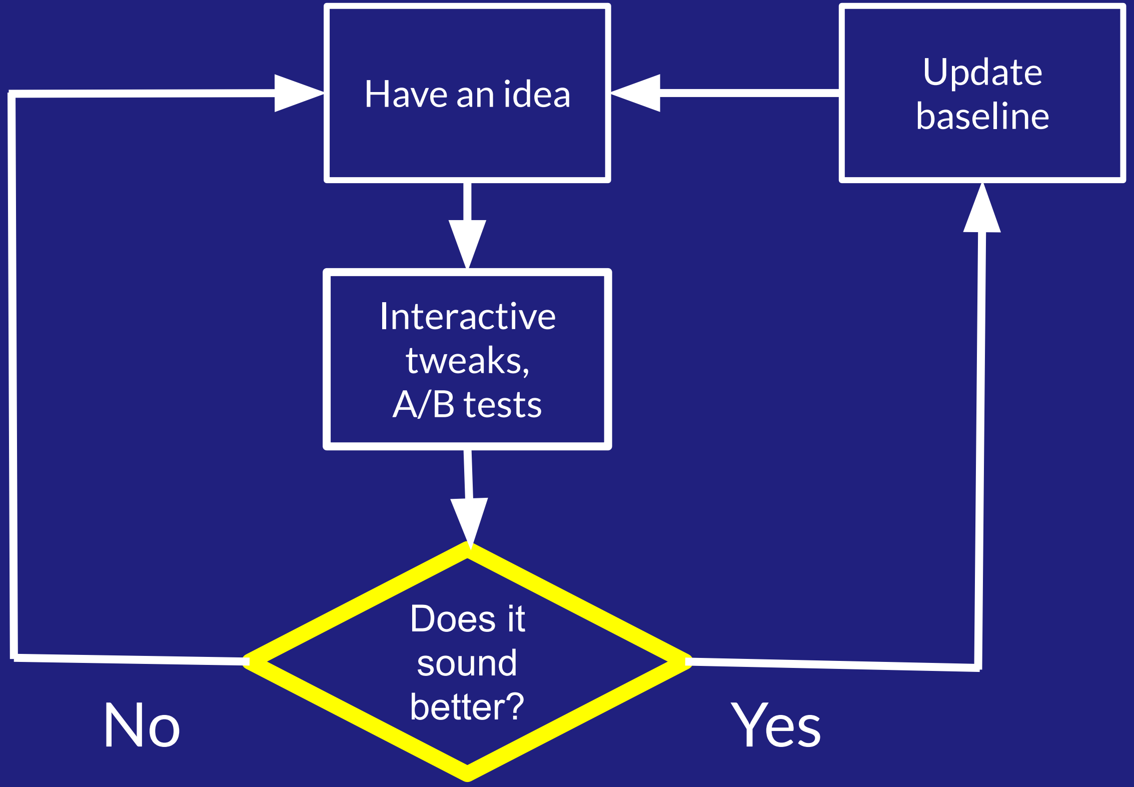
- Sometimes need to kill bad concepts and step back in sound quality.

- Some constants are “covering up” bad values of other constants

A sort of manual gradient descent

- Need to retune already tuned constants.
- Sometimes need to kill bad concepts and step back in sound quality.
- Some constants are “covering up” bad values of other constants
- Human audition degrades with age

Replace this step?



Have an idea

Interactive tweaks, A/B tests

Does it sound better?

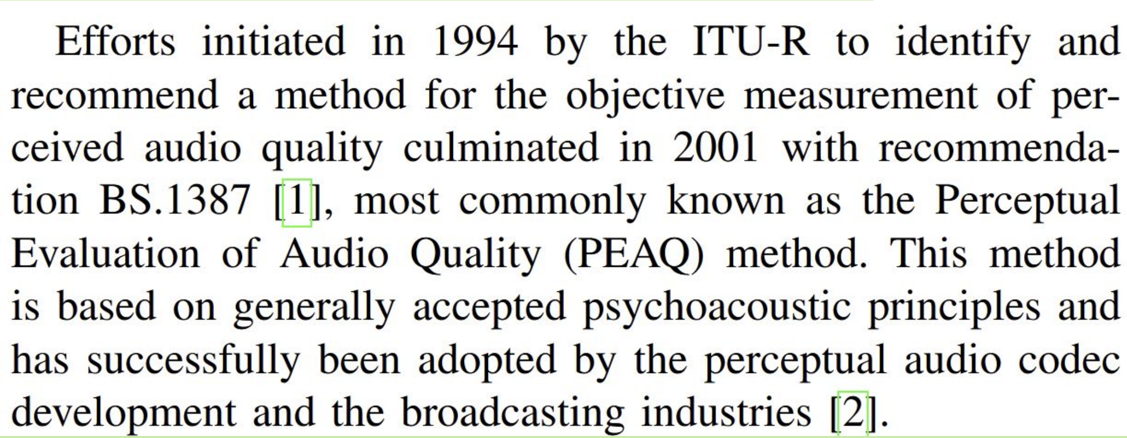
No

Yes

Update baseline

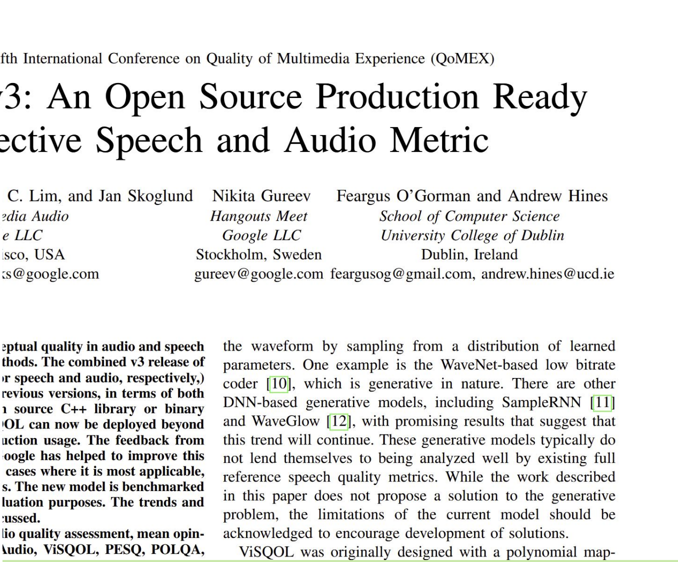
We are living in the future.

OPTICOM
THE PERCEPTUAL QUALITY EXPERTS



Efforts initiated in 1994 by the ITU-R to identify and recommend a method for the objective measurement of perceived audio quality culminated in 2001 with recommendation BS.1387 [1], most commonly known as the Perceptual Evaluation of Audio Quality (PEAQ) method. This method is based on generally accepted psychoacoustic principles and has successfully been adopted by the perceptual audio codec development and the broadcasting industries [2].

PEAQ (1998)



2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)

ViSQOL v3: An Open Source Production Ready Objective Speech and Audio Metric

C. Lim, and Jan Skoglund
edia Audio
e LLC
sco, USA
s@google.com

Nikita Gureev
Hangouts Meet
Google LLC
Stockholm, Sweden
gureev@google.com

Feargus O'Gorman and Andrew Hines
School of Computer Science
University College of Dublin
Dublin, Ireland
feargusog@gmail.com, andrew.hines@ucd.ie

**eptual quality in audio and speech
thods. The combined v3 release of
r speech and audio, respectively,)
revious versions, in terms of both
source C++ library or binary
OL can now be deployed beyond
uction usage. The feedback from
oogle has helped to improve this
cases where it is most applicable,
s. The new model is benchmarked
luation purposes. The trends and
ussed.**

**io quality assessment, mean opin-
Audio, ViSQOL, PESQ, POLQA,**

the waveform by sampling from a distribution of learned parameters. One example is the WaveNet-based low bitrate coder [10], which is generative in nature. There are other DNN-based generative models, including SampleRNN [11] and WaveGlow [12], with promising results that suggest that this trend will continue. These generative models typically do not lend themselves to being analyzed well by existing full reference speech quality metrics. While the work described in this paper does not propose a solution to the generative problem, the limitations of the current model should be acknowledged to encourage development of solutions.

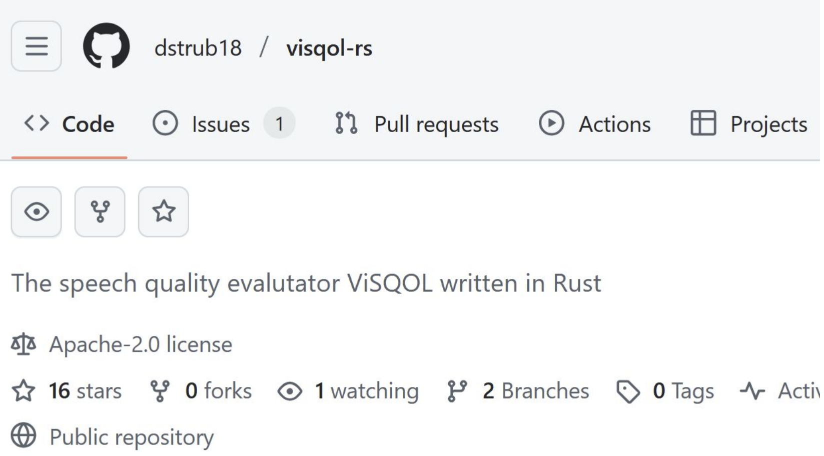
ViSQOL was originally designed with a polynomial map-

ViSQOL v3 (2020)

Perceptual objective measures are a thing

Settled on **visqol-rs**

- written in **Rust**
- based upon Google research and **AI** model



dstrub18 / visqol-rs

The speech quality evalutator ViSQOL written in Rust

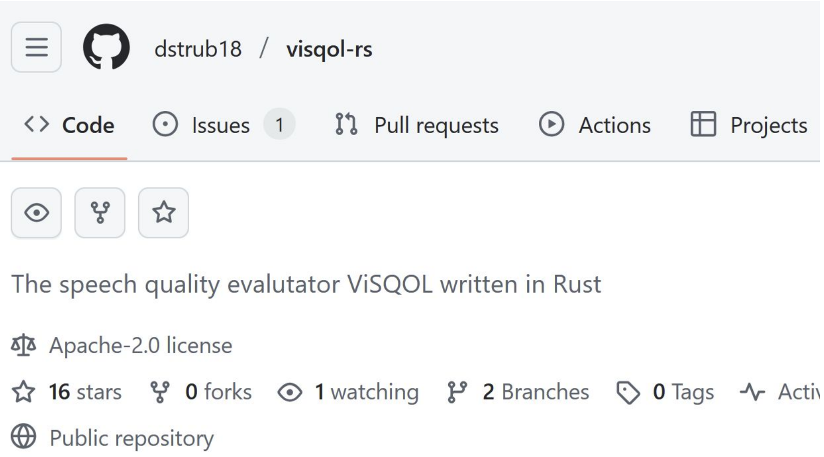
Apache-2.0 license

16 stars 0 forks 1 watching 2 Branches 0 Tags Activ

Public repository

Settled on **visqol-rs**

- written in **Rust**
- based upon Google research and **AI** model



dstrub18 / visqol-rs

The speech quality evalutator ViSQOL written in Rust

Apache-2.0 license

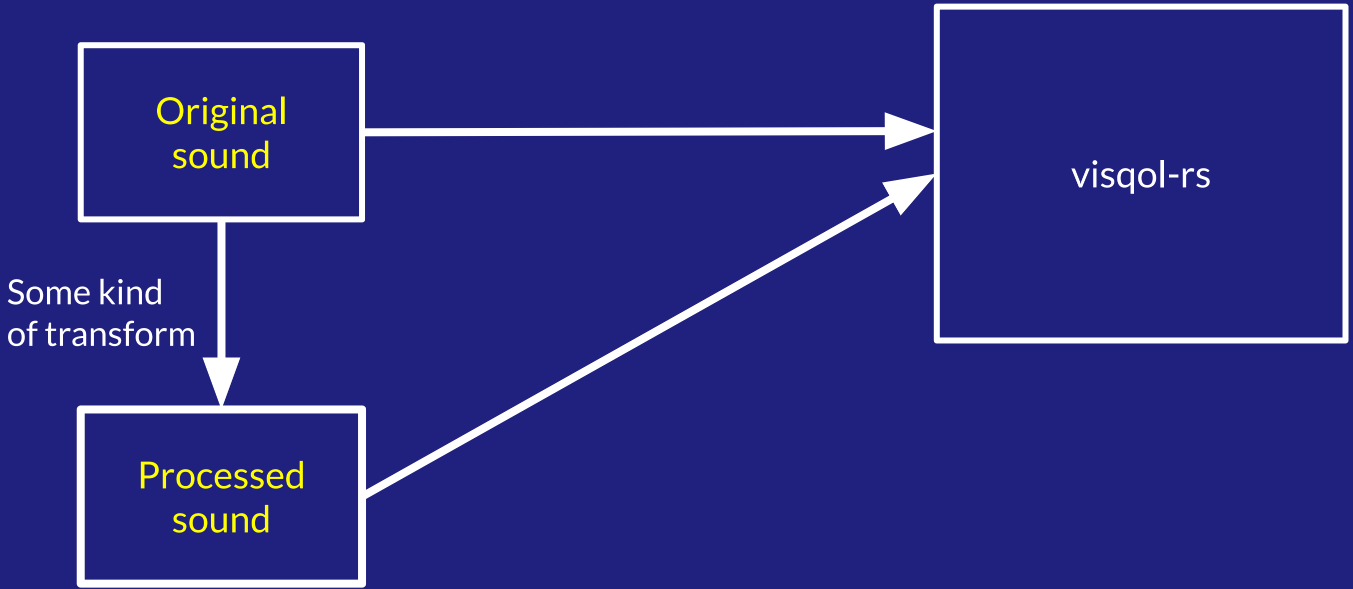
16 stars 0 forks 1 watching 2 Branches 0 Tags Activ

Public repository

My stuff is finally running through a neural network!

How would a machine know what is “good sound”?

Audio objective measures

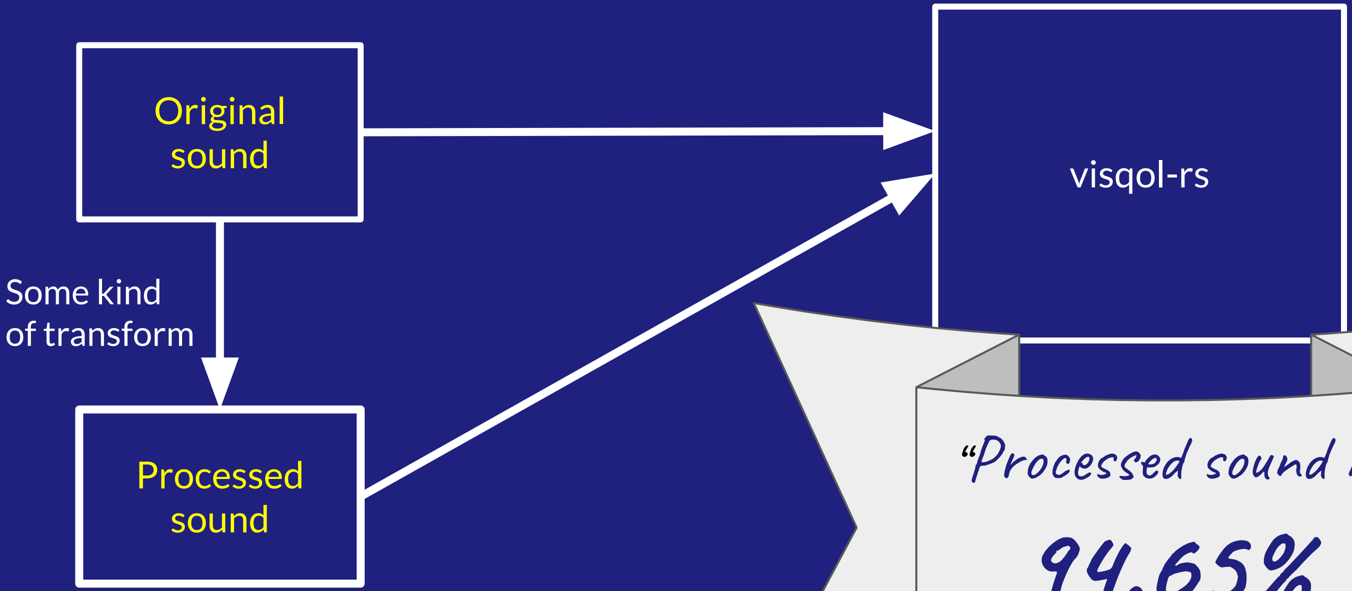

Original sound

Some kind of transform

Processed sound

visqol-rs

Audio objective measures



Original sound

Some kind of transform

Processed sound

visqol-rs

"Processed sound is

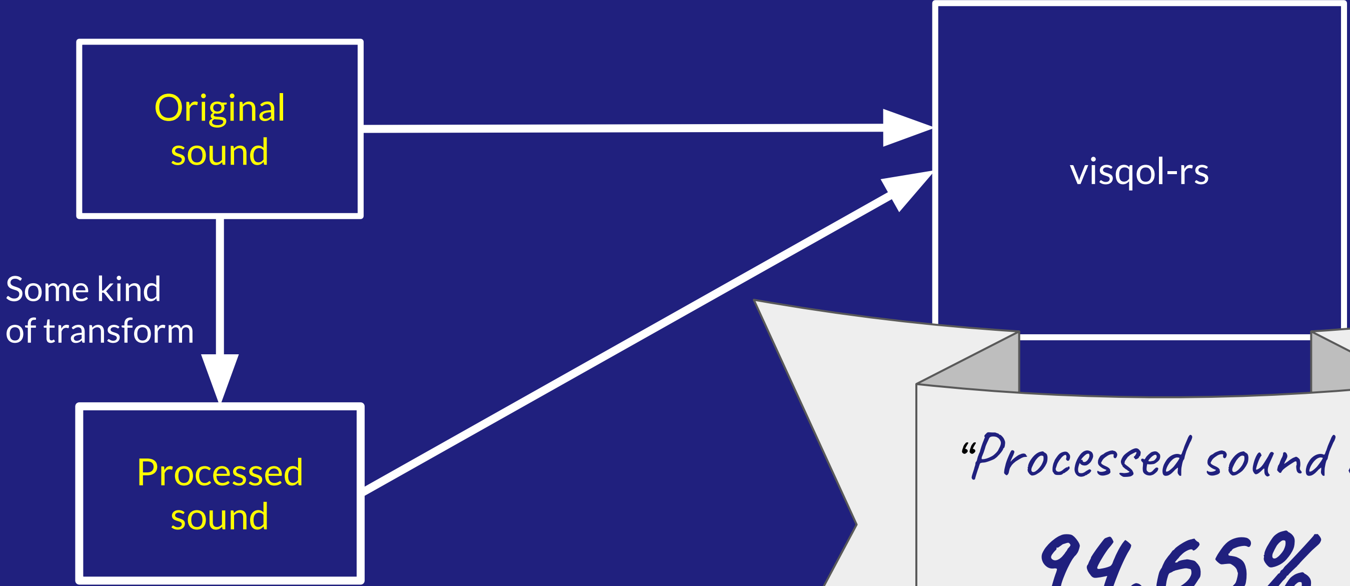
94.65%

like the original, according to AI."

That, I evaluated.

Audio objective measures

That, I evaluated.



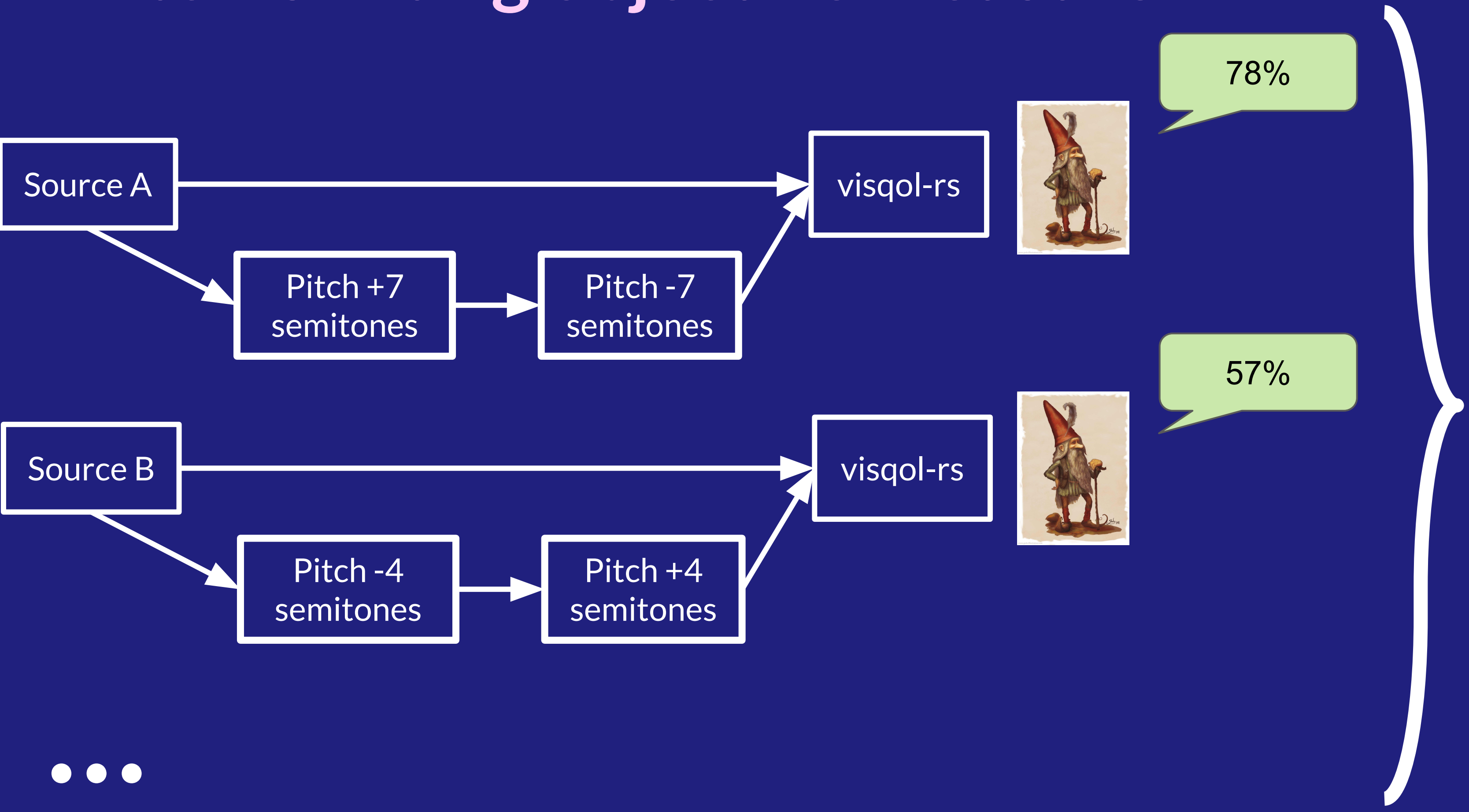
"Processed sound is

94.65%

like the original, according to AI."

In the codec sense, quality is meant as reproduction.

Pitch-shifting objective measure



Source A

Pitch +7 semitones

Pitch -7 semitones

visqol-rs

78%

Source B

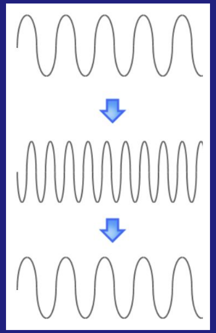
Pitch -4 semitones

Pitch +4 semitones

visqol-rs

57%

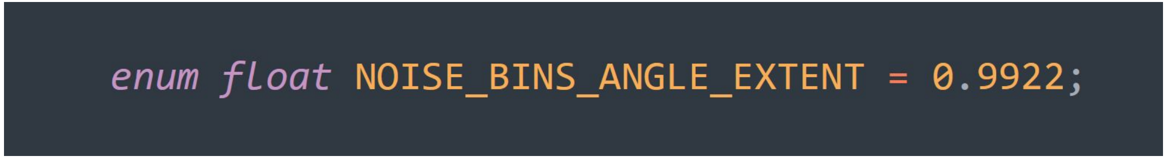
Weighted sum



•••

Yes, but…

You’re going to want to modify all my magic constants, right?



```
enum float NOISE_BINS_ANGLE_EXTENT = 0.9922;
```

You’re going to like… modify all my magic constants, right?

Well, yeah. That’s what optimization does.

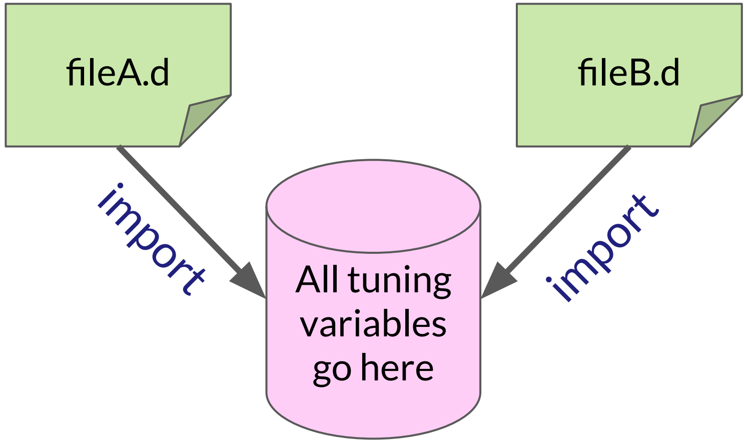
constants, right?

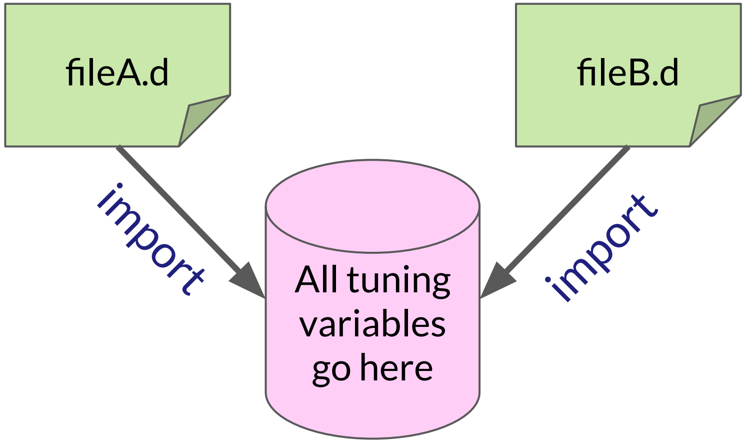
Well, yeah. That’s what optimization does.

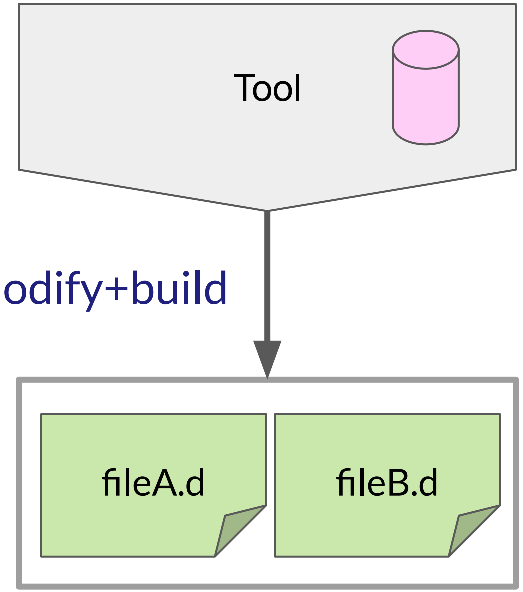
Take a look at the codebase. There is no way to put them all in one place.

Take a loot at the codebase. There is no way to put them all in one place.

I see. We'll find a way to just add some UDAs.


fileA.d
fileB.d
import
import
All tuning variables go here

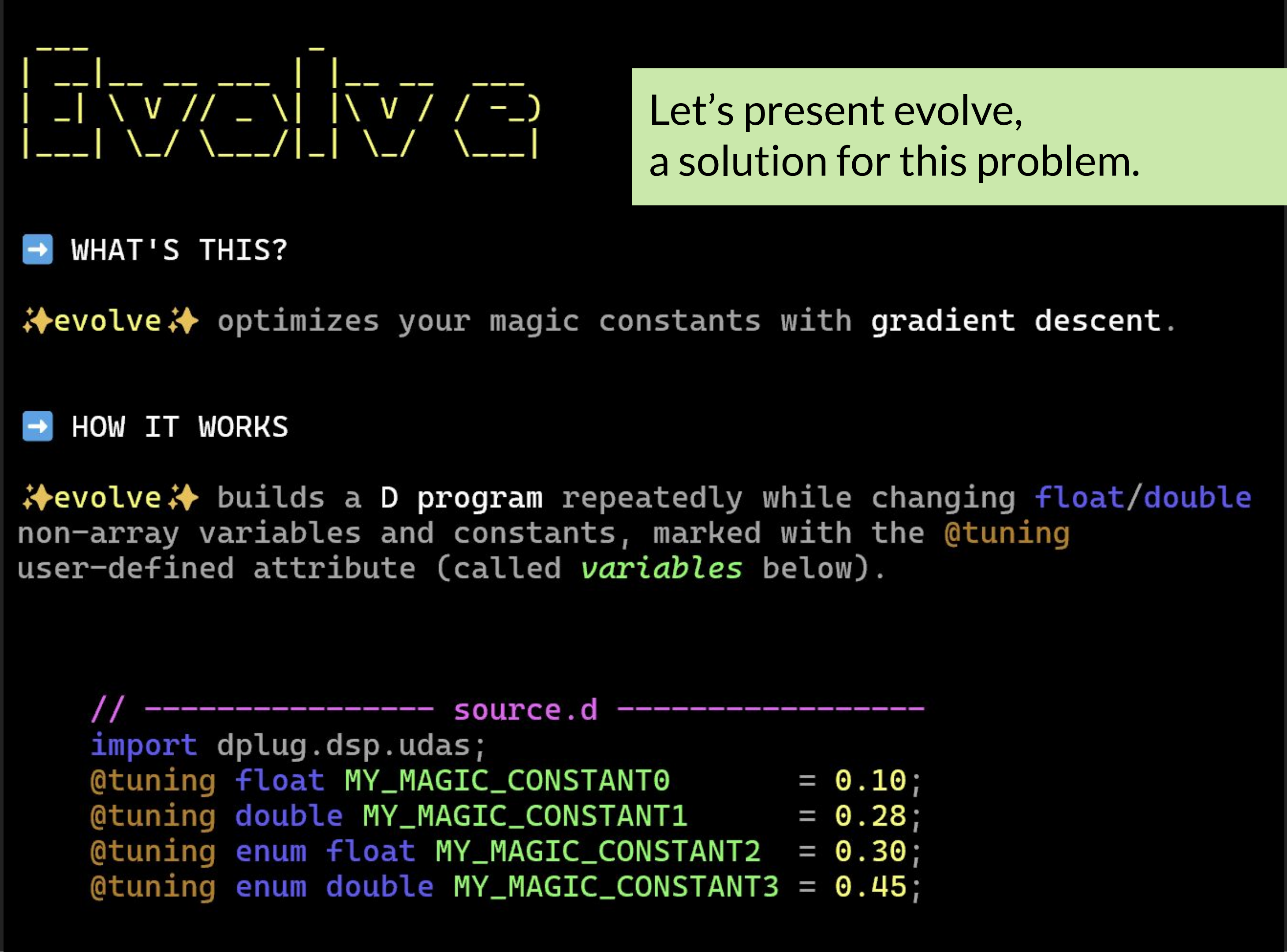




Instead of gathering all parameters in one single place:

- 1. Parse source file
- 2. Regenerate source code
- 3. Rebuild and evaluate

Let’s present evolve,
a solution for this problem.



```
 ___             _
| __|__ __ ___ | |__ __ ___
| _| \ V // _ \| |\ V / / -_)
|___| \_/ \___/|_| \_/  \___|
```

➡️ WHAT'S THIS?

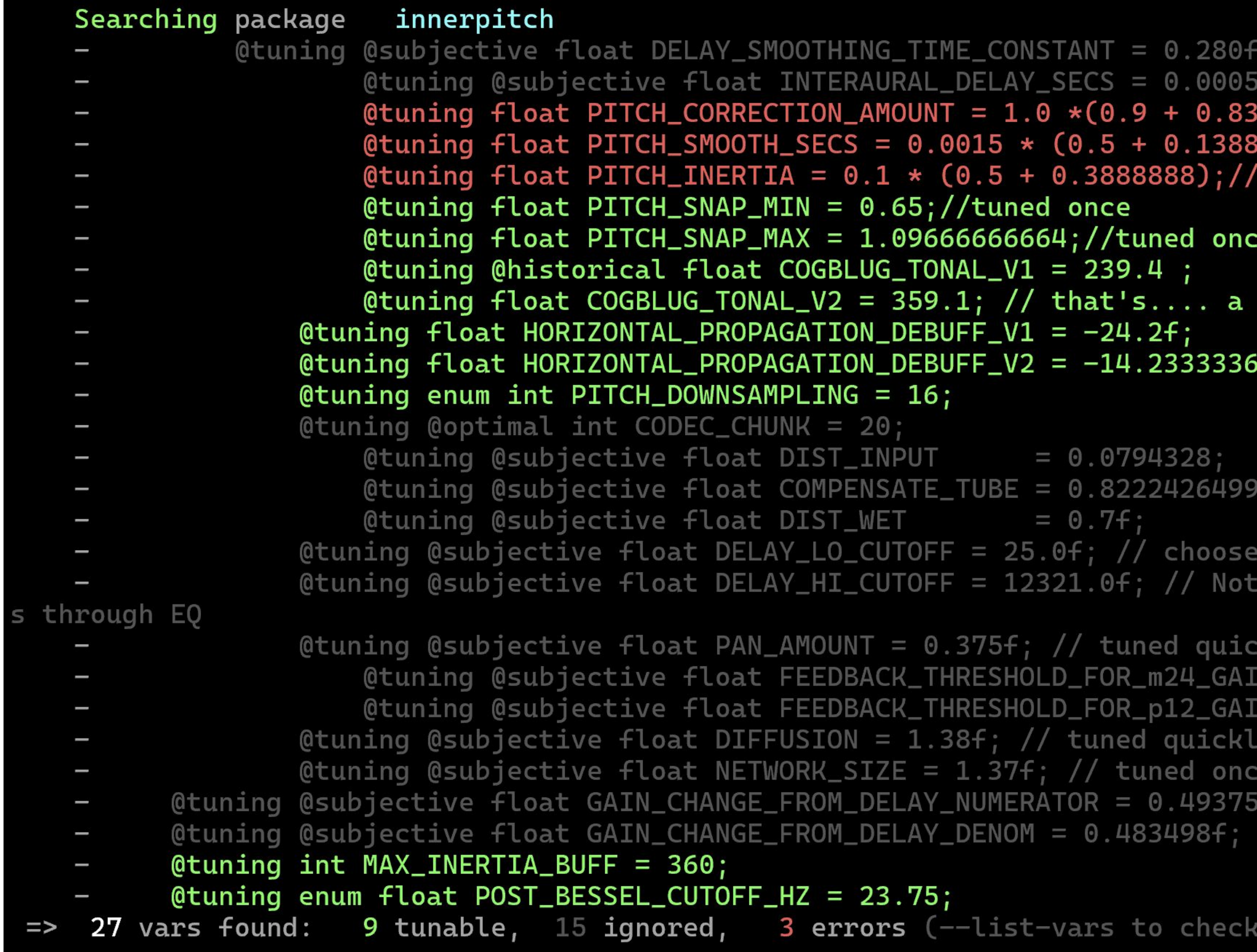
✨evolve✨ optimizes your magic constants with **gradient descent**.

➡️ HOW IT WORKS

✨evolve✨ builds a **D program** repeatedly while changing float/double non-array variables and constants, marked with the @tuning user-defined attribute (called *variables* below).

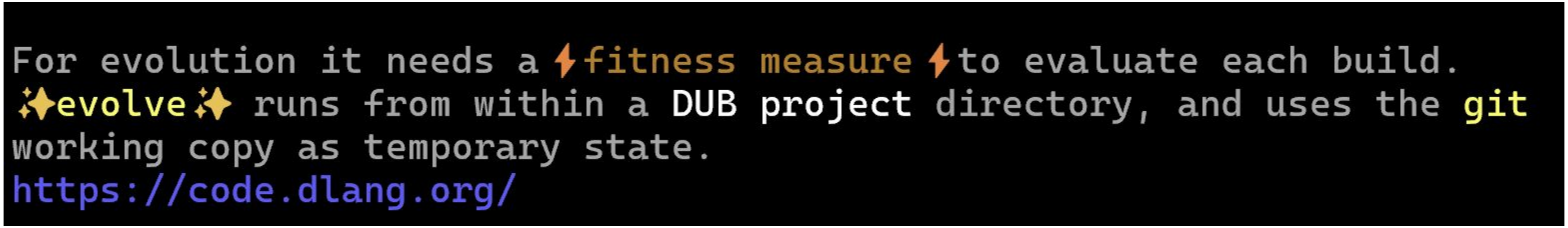
```
// ---------------- source.d -----------------
import dplug.dsp.udas;
@tuning float MY_MAGIC_CONSTANT0        = 0.10;
@tuning double MY_MAGIC_CONSTANT1       = 0.28;
@tuning enum float MY_MAGIC_CONSTANT2   = 0.30;
@tuning enum double MY_MAGIC_CONSTANT3 = 0.45;
```

evolve tool can list tuning variables and has semantic UDAs for ignoring some if needed.



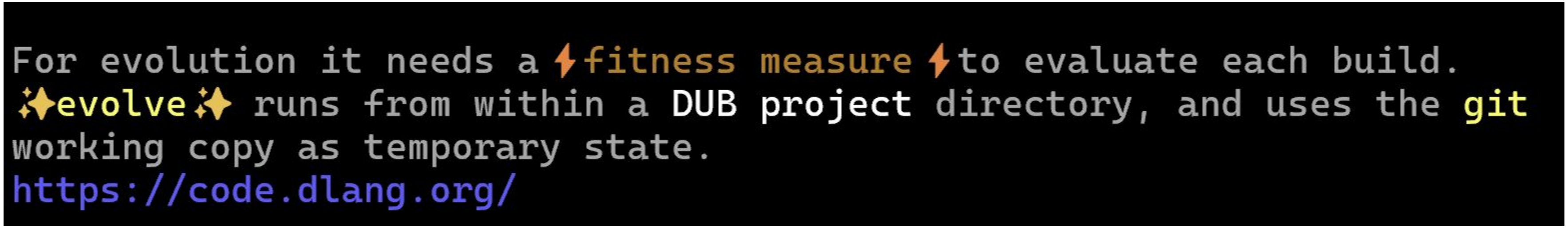
```
    Searching package   innerpitch
    -         @tuning @subjective float DELAY_SMOOTHING_TIME_CONSTANT = 0.280f
    -                 @tuning @subjective float INTERAURAL_DELAY_SECS = 0.0005
    -                 @tuning float PITCH_CORRECTION_AMOUNT = 1.0 *(0.9 + 0.83
    -                 @tuning float PITCH_SMOOTH_SECS = 0.0015 * (0.5 + 0.1388
    -                 @tuning float PITCH_INERTIA = 0.1 * (0.5 + 0.3888888);//
    -                 @tuning float PITCH_SNAP_MIN = 0.65;//tuned once
    -                 @tuning float PITCH_SNAP_MAX = 1.09666666664;//tuned onc
    -                 @tuning @historical float COGBLUG_TONAL_V1 = 239.4 ;
    -                 @tuning float COGBLUG_TONAL_V2 = 359.1; // that's.... a
    -             @tuning float HORIZONTAL_PROPAGATION_DEBUFF_V1 = -24.2f;
    -             @tuning float HORIZONTAL_PROPAGATION_DEBUFF_V2 = -14.2333336
    -             @tuning enum int PITCH_DOWNSAMPLING = 16;
    -             @tuning @optimal int CODEC_CHUNK = 20;
    -                 @tuning @subjective float DIST_INPUT      = 0.0794328;
    -                 @tuning @subjective float COMPENSATE_TUBE = 0.8222426499
    -                 @tuning @subjective float DIST_WET        = 0.7f;
    -             @tuning @subjective float DELAY_LO_CUTOFF = 25.0f; // choose
    -             @tuning @subjective float DELAY_HI_CUTOFF = 12321.0f; // Not
s through EQ
    -             @tuning @subjective float PAN_AMOUNT = 0.375f; // tuned quic
    -                 @tuning @subjective float FEEDBACK_THRESHOLD_FOR_m24_GAI
    -                 @tuning @subjective float FEEDBACK_THRESHOLD_FOR_p12_GAI
    -             @tuning @subjective float DIFFUSION = 1.38f; // tuned quickl
    -             @tuning @subjective float NETWORK_SIZE = 1.37f; // tuned onc
    -     @tuning @subjective float GAIN_CHANGE_FROM_DELAY_NUMERATOR = 0.49375
    -     @tuning @subjective float GAIN_CHANGE_FROM_DELAY_DENOM = 0.483498f;
    -     @tuning int MAX_INERTIA_BUFF = 360;
    -     @tuning enum float POST_BESSEL_CUTOFF_HZ = 23.75;
 =>  27 vars found:   9 tunable,  15 ignored,   3 errors (--list-vars to check
```

Bring your own fitness measure.

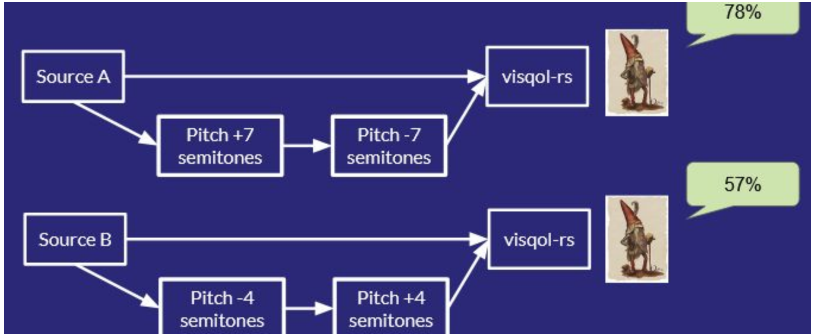


```
For evolution it needs a ⚡fitness measure⚡to evaluate each build.
✨evolve✨ runs from within a DUB project directory, and uses the git
working copy as temporary state.
https://code.dlang.org/
```

Bring your own fitness measure.

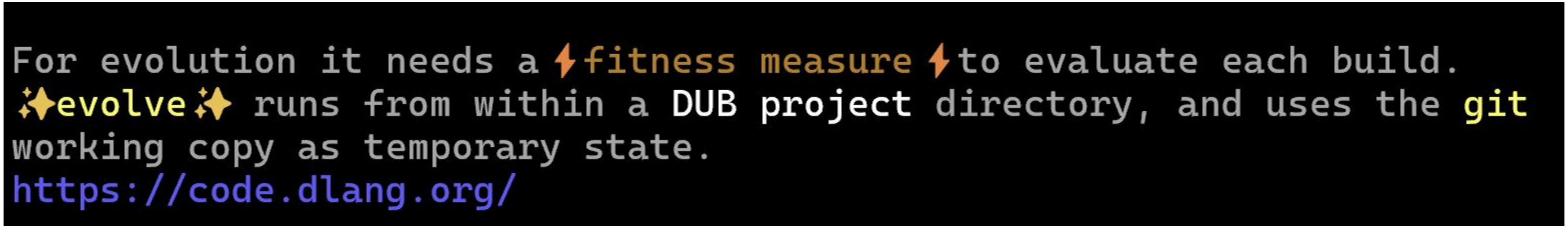


```
For evolution it needs a ⚡fitness measure⚡to evaluate each build.
✨evolve✨ runs from within a DUB project directory, and uses the git
working copy as temporary state.
https://code.dlang.org/
```



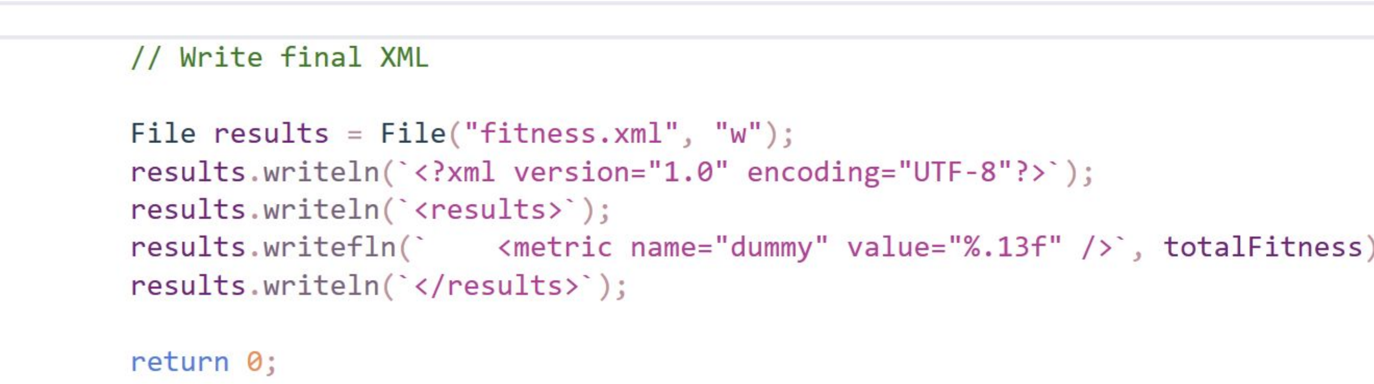
In pitch-shifting case

Bring your own fitness measure.

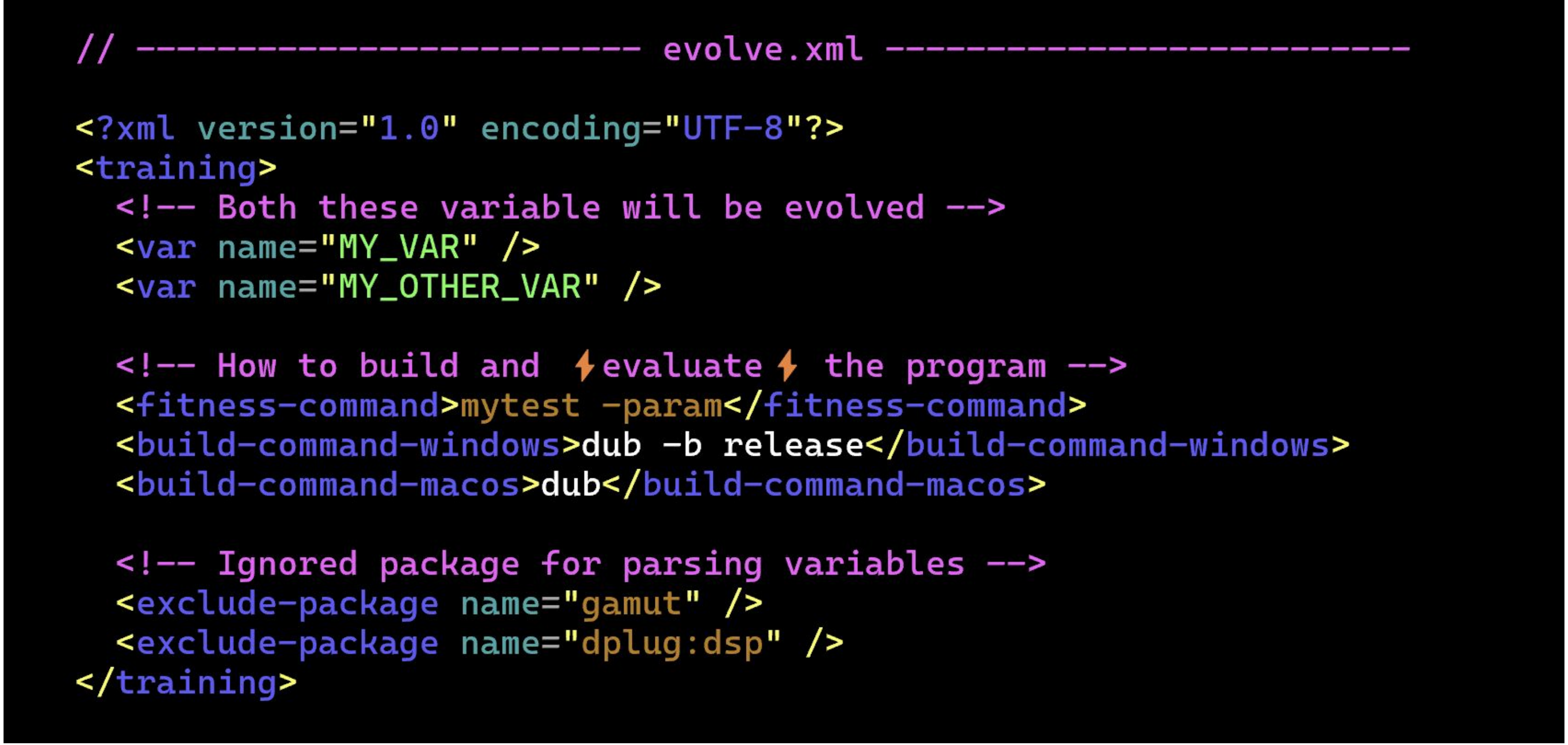


```
For evolution it needs a ⚡fitness measure⚡to evaluate each build.
✨evolve✨ runs from within a DUB project directory, and uses the git
working copy as temporary state.
https://code.dlang.org/
```

Fitness program must return a **fitness.xml** file with one number.



```
// Write final XML

File results = File("fitness.xml", "w");
results.writeln(`<?xml version="1.0" encoding="UTF-8"?>`);
results.writeln(`<results>`);
results.writefln(`    <metric name="dummy" value="%.13f" />`, totalFitness)
results.writeln(`</results>`);

return 0;
```

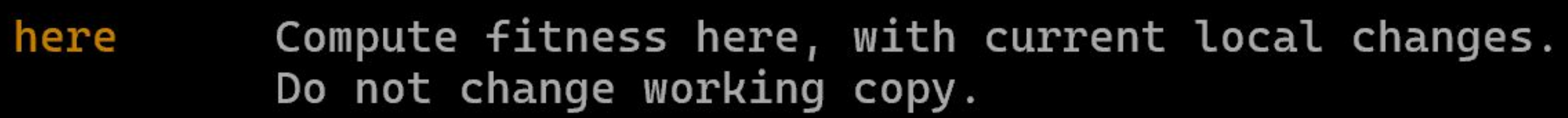


```
// ------------------------ evolve.xml -------------------------

<?xml version="1.0" encoding="UTF-8"?>
<training>
  <!-- Both these variable will be evolved -->
  <var name="MY_VAR" />
  <var name="MY_OTHER_VAR" />

  <!-- How to build and ⚡evaluate⚡ the program -->
  <fitness-command>mytest -param</fitness-command>
  <build-command-windows>dub -b release</build-command-windows>
  <build-command-macos>dub</build-command-macos>

  <!-- Ignored package for parsing variables -->
  <exclude-package name="gamut" />
  <exclude-package name="dplug:dsp" />
</training>
```

Configuration file.

Tool here: https://github.com/AuburnSounds/Dplug/tree/master/tools/evolve

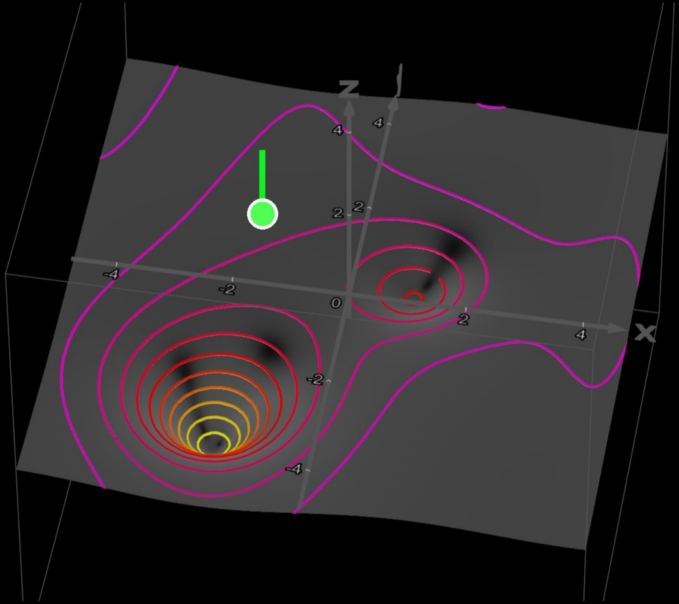
4 search algorithms in evolve tool



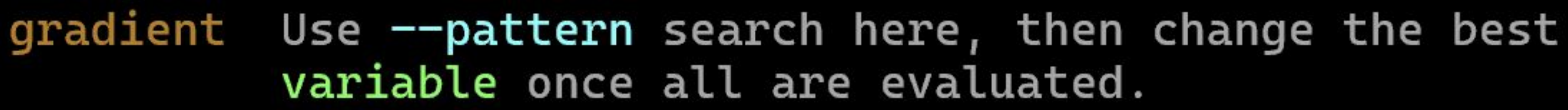
```
here        Compute fitness here, with current local changes.
            Do not change working copy.
```

```
evolve -a here
```

= Just displays current fitness.

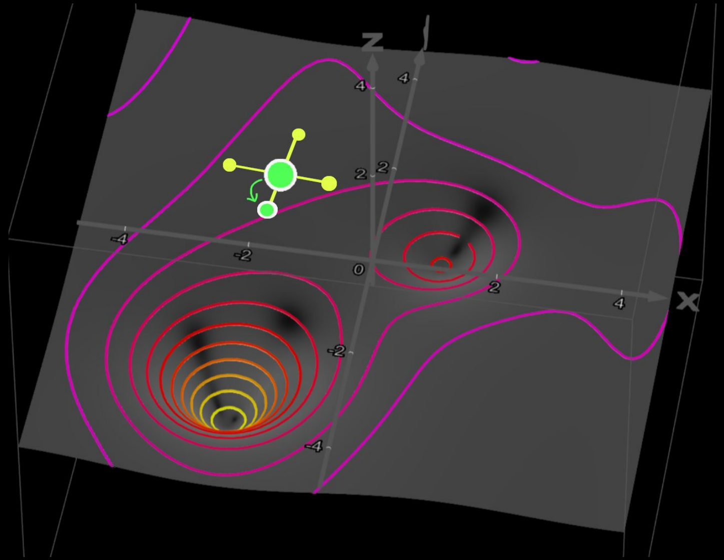


4 search algorithms in evolve tool



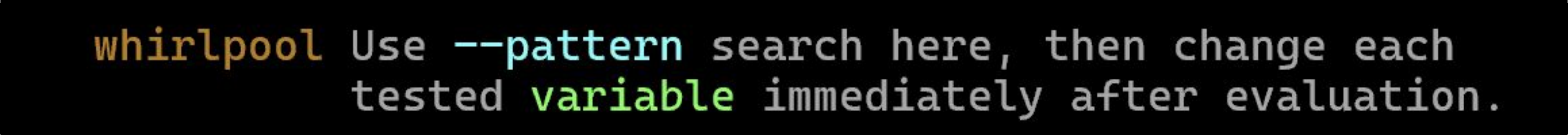
```
gradient  Use --pattern search here, then change the best
          variable once all are evaluated.
```

Pattern search in each dimension. Pick single best improvement.

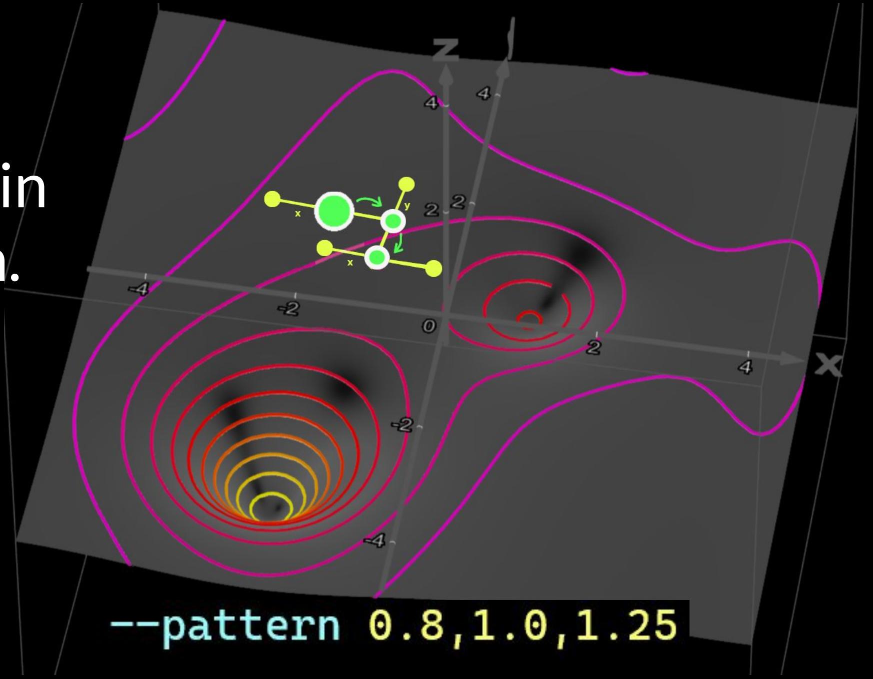


```
--pattern 0.8,1.0,1.25
```

4 search algorithms in evolve tool



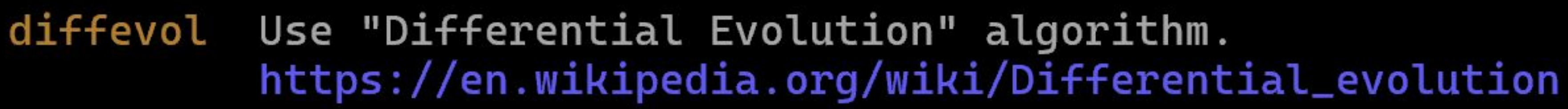
```
whirlpool Use --pattern search here, then change each
          tested variable immediately after evaluation.
```



```
--pattern 0.8,1.0,1.25
```

Pattern search in each dimension. Change immediately to better value.

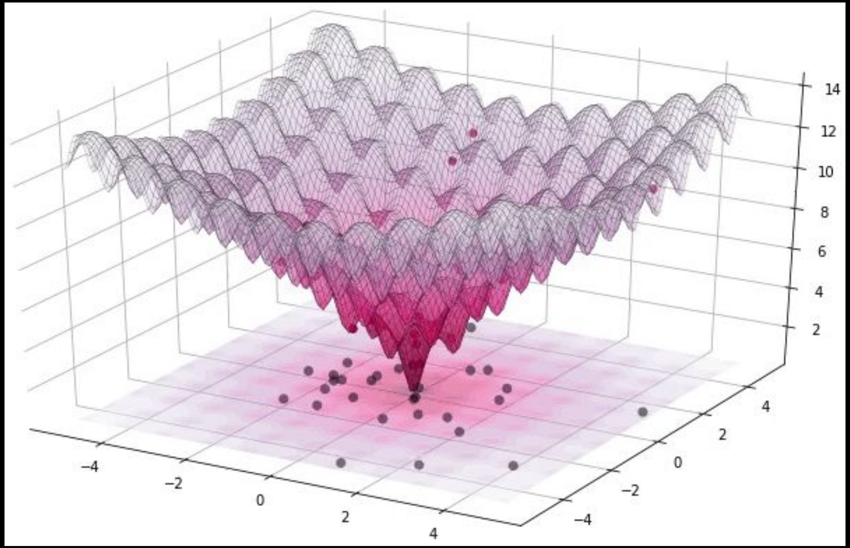
4 search algorithms in evolve tool



```
diffevol   Use "Differential Evolution" algorithm.
           https://en.wikipedia.org/wiki/Differential_evolution
```

A famously simple metaheuristic optimization algorithm.

Typical population > 15

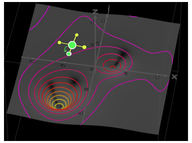


(source = https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/)

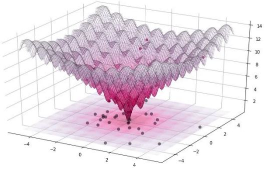
QUIZZ QUESTION

What will be the most useful algorithm in practice?

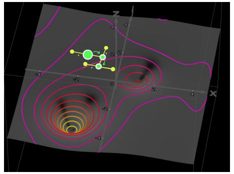
- A. gradient



- B. differential evolution



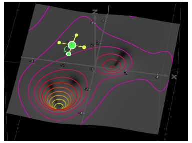
- C. whirlpool



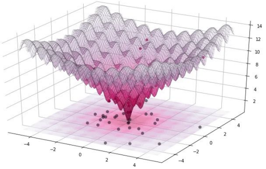
QUIZZ ANSWER

What will be the most useful algorithm in practice?

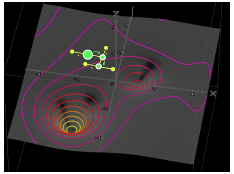
- A. gradient



- B. differential evolution

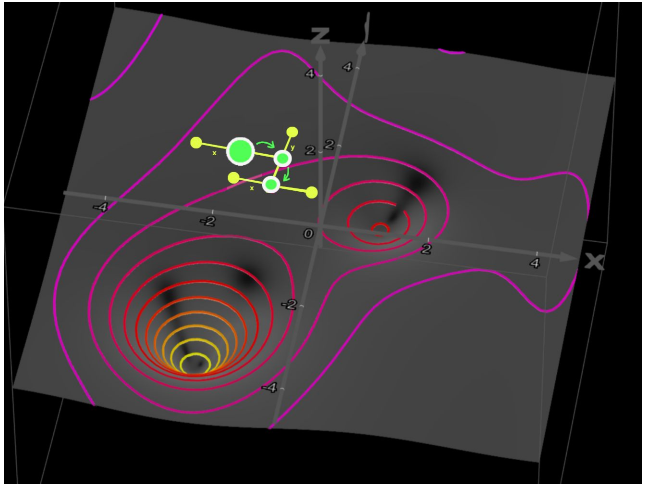


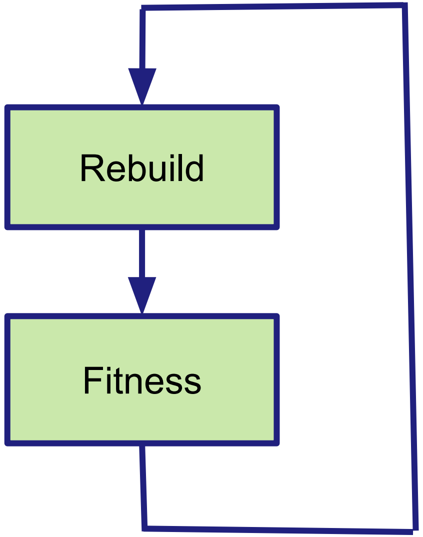
- **C. whirlpool**



Build times are pretty slow

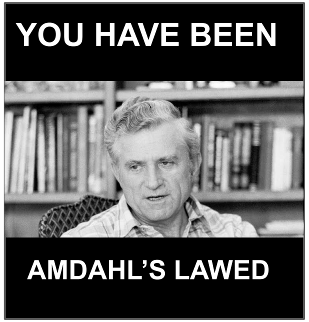
“whirlpool” method just makes less evaluations and **move on.**

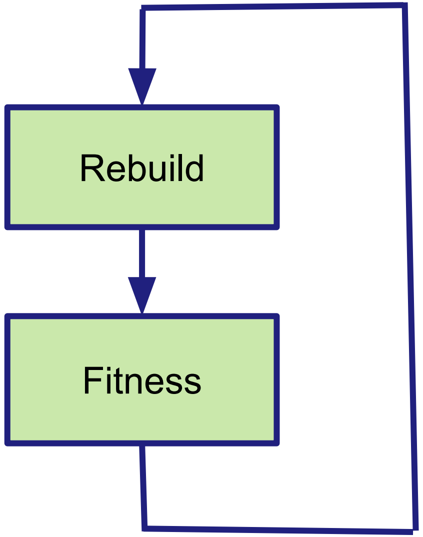




Build times are pretty slow

- Fitness evaluation may be fast, but it doesn’t matter since rebuilding is rather slow.
- Might as well have a **slow fitness evalution**

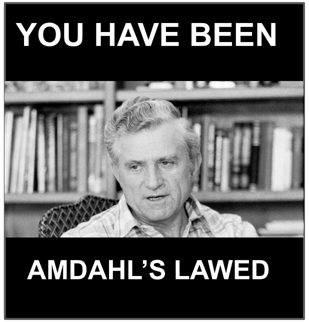




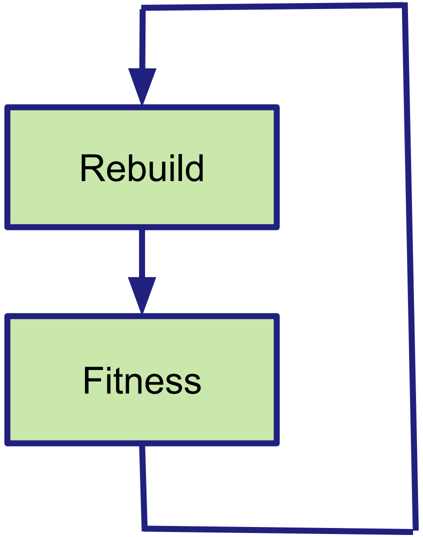
Build times are pretty slow

- Fitness evaluation may be fast, but it doesn't matter since rebuilding is rather slow.

- Might as well have a **slow fitness evalution**



The evolve tool is applicable
where the fitness evaluation
is slow,
such as perceptual measures.

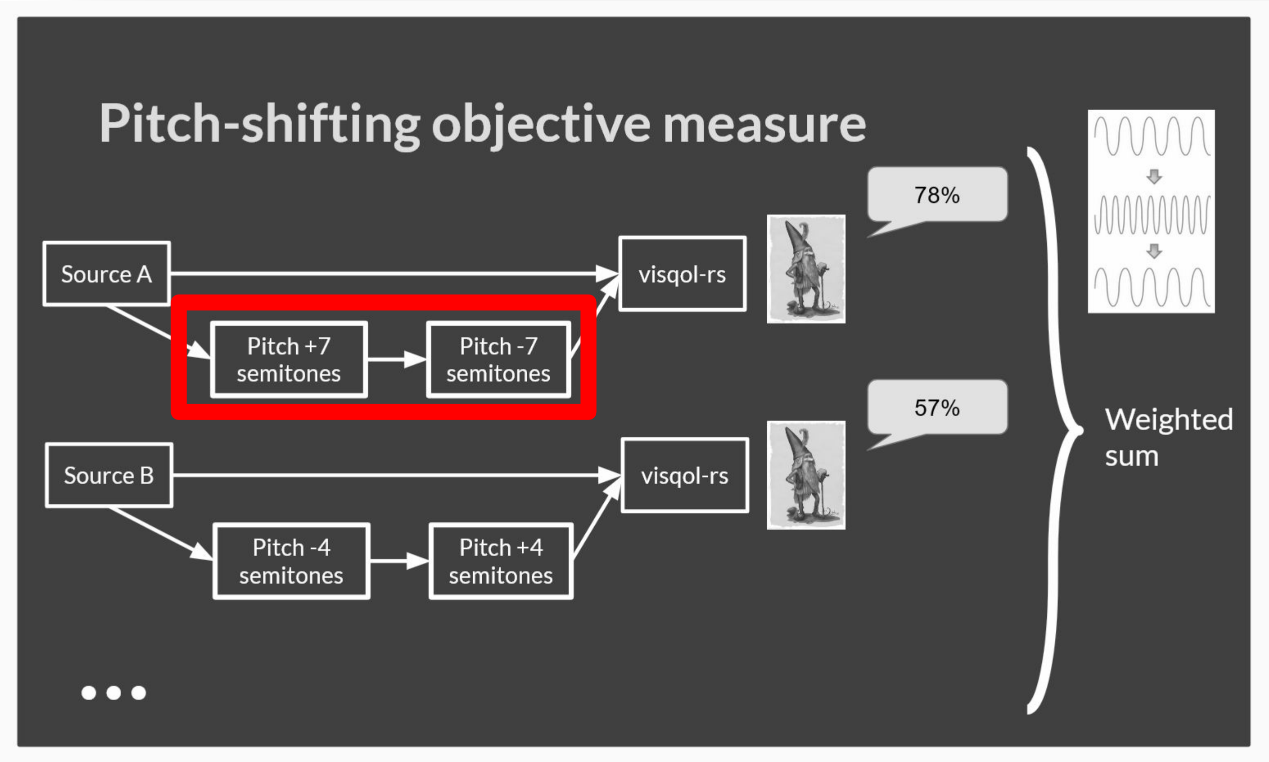


Just 26 man-days after starting the automatic optimization effort, Just 26 man-days after starting the automatic optimization effort, sound quality had actually decreased.

📉

Everything that went wrong

A. Remember slide 33?



Pitch-shifting objective measure

Source A

Pitch +7 semitones

Pitch -7 semitones

visqol-rs

78%

Source B

Pitch -4 semitones

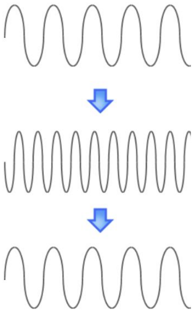
Pitch +4 semitones

visqol-rs

57%

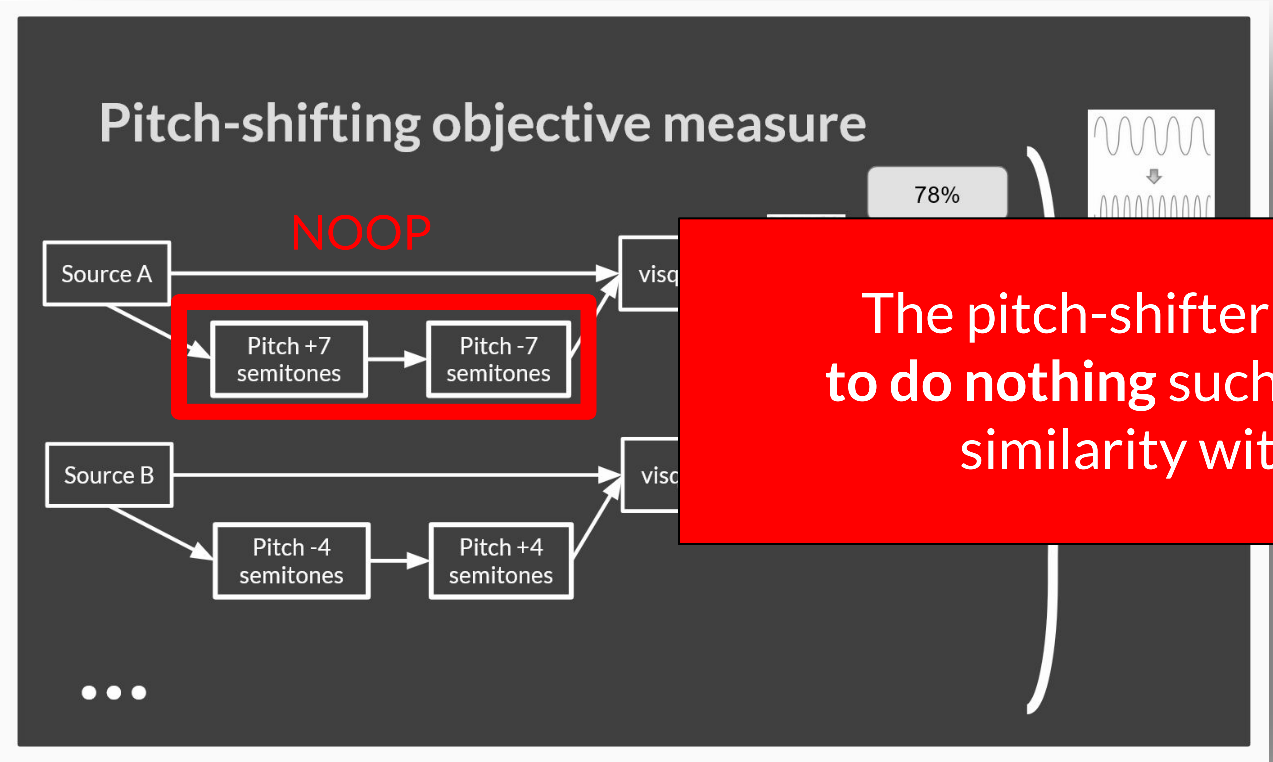
Weighted sum

•••

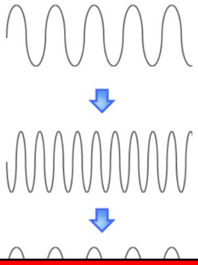


We proposed to shift +N then -N to return close to the original sound.

A. Remember slide 33?

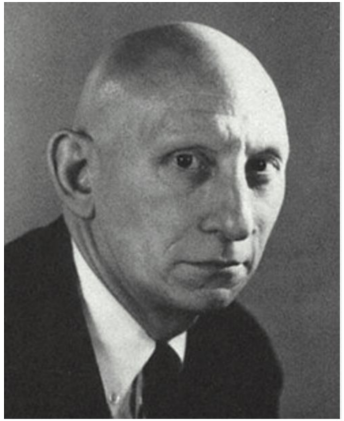


Pitch-shifting objective measure



The pitch-shifter quickly learnt **to do nothing** such as to maximize similarity with original.

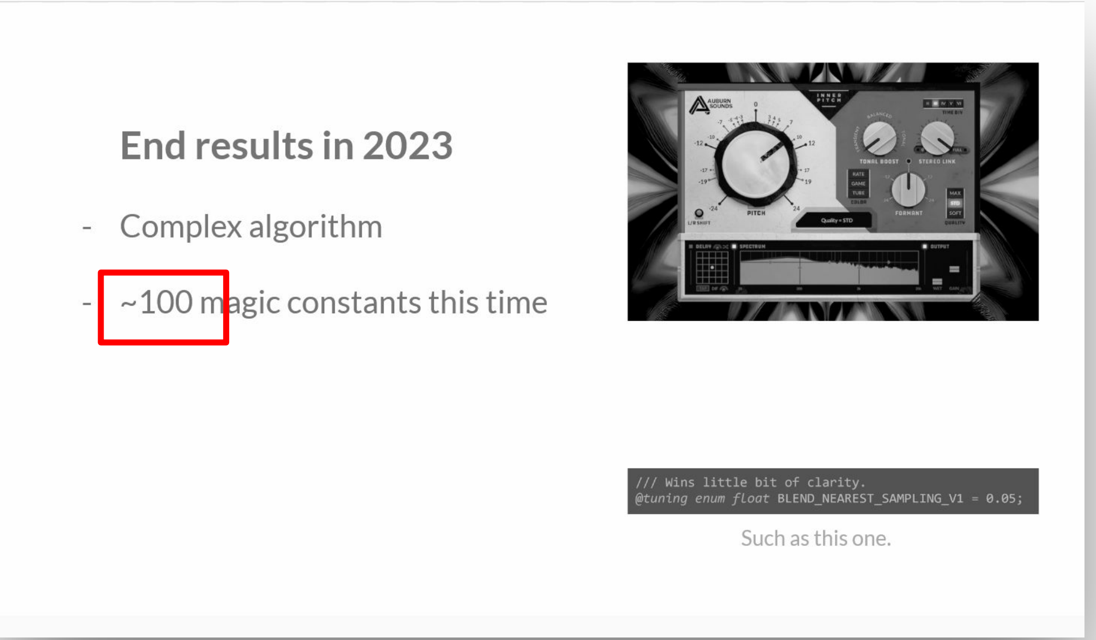
then -N
to return close
to the original
sound.



When we write programs
that "learn", it turns out
that we do and they don't.

— Alan Perlis

B. Remember slide 19?



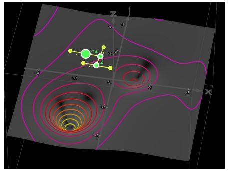
End results in 2023

- Complex algorithm
- ~100 magic constants this time

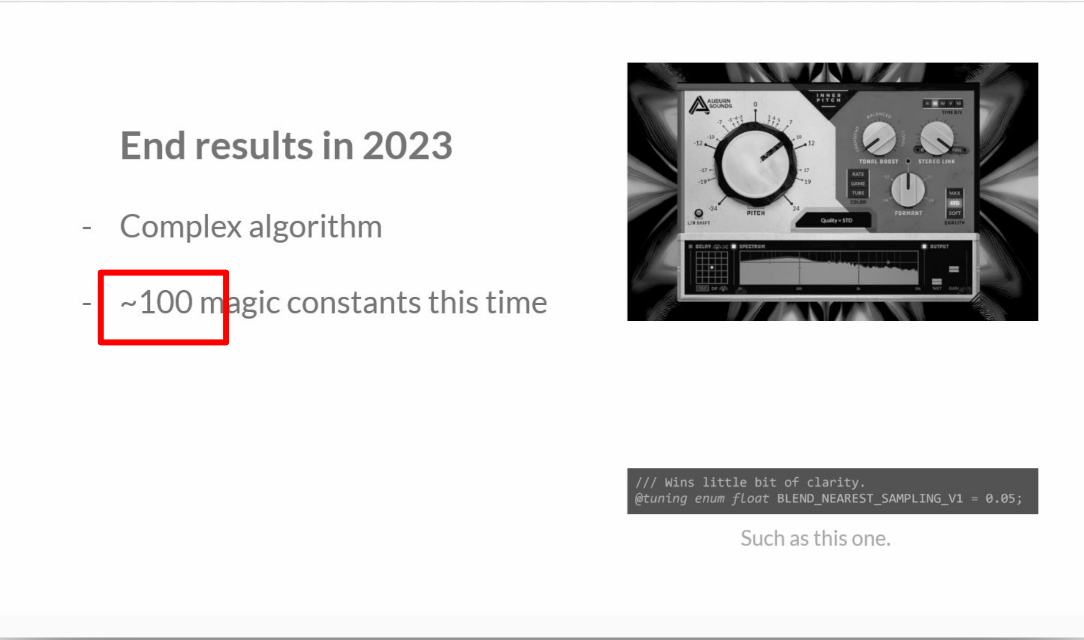
```
/// Wins little bit of clarity.
@tuning enum float BLEND_NEAREST_SAMPLING_V1 = 0.05;
```

Such as this one.

We said to have many parameters to evolve, and that whirlpool was used.



B. Remember slide 19?



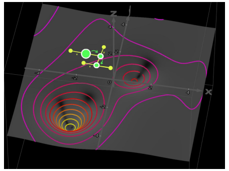
End results in 2023

- Complex algorithm
- ~100 magic constants this time

```
/// Wins little bit of clarity.
@tuning enum float BLEND_NEAREST_SAMPLING_V1 = 0.05;
```

Such as this one.

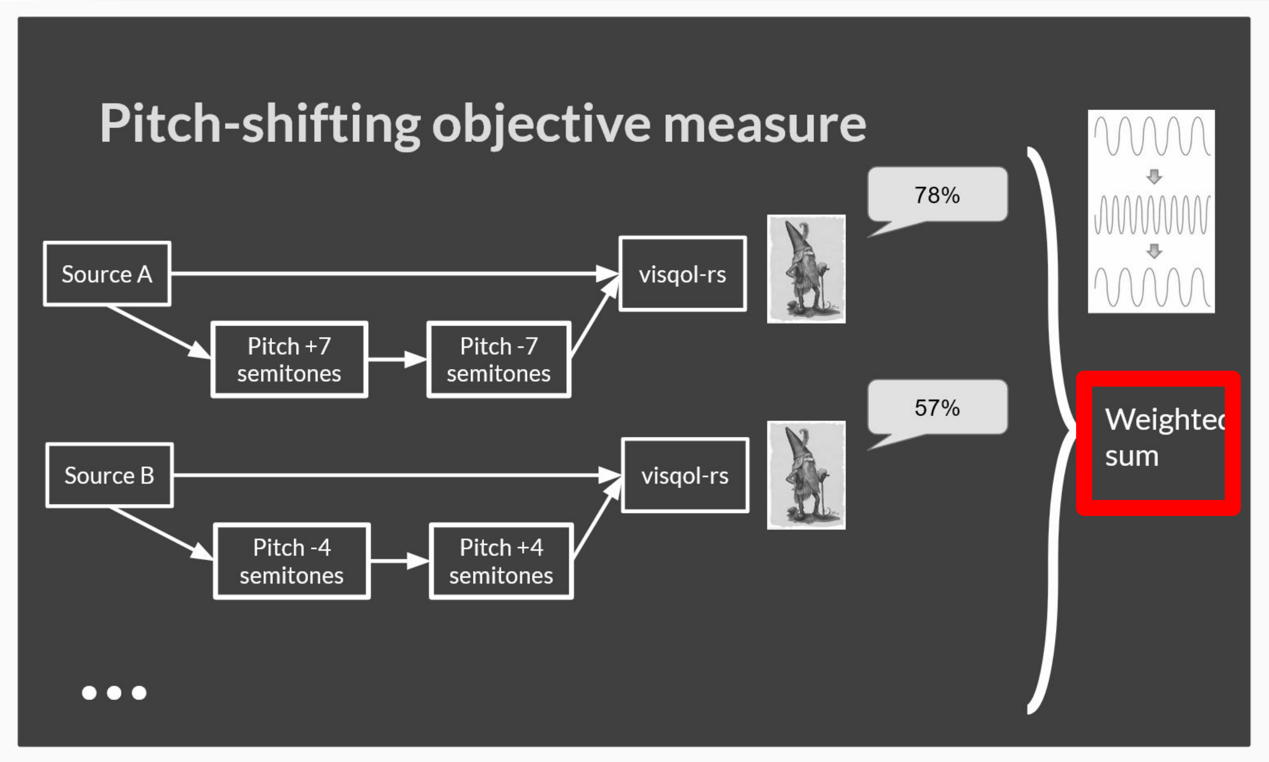
We said to have many parameters to evolve, and that whirlpool was used.



HIGH DIMENSIONS ARE NOT ADVISED

Easy to make minor “progress” indefinitely.

C. Again slide 33



Pitch-shifting objective measure

Source A

Pitch +7 semitones

Pitch -7 semitones

visqol-rs

78%

Source B

Pitch -4 semitones

Pitch +4 semitones

visqol-rs

57%

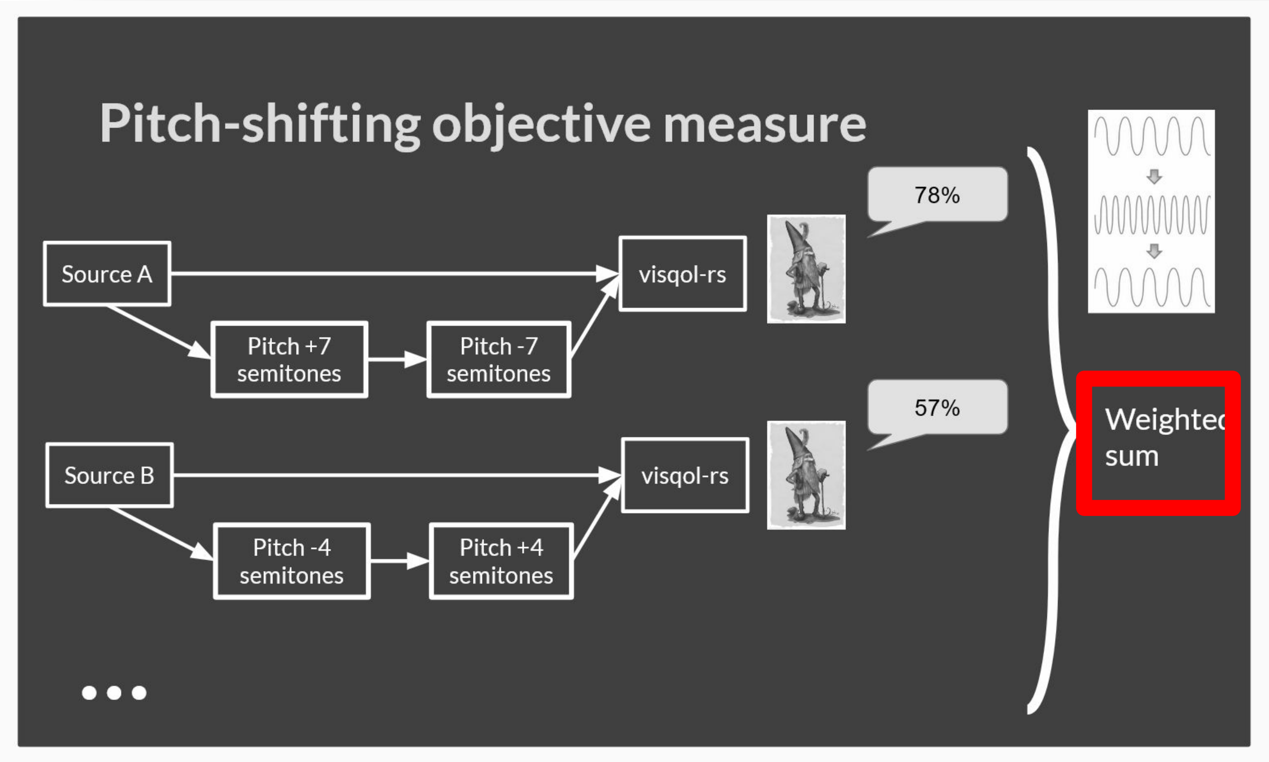
Weightec sum

•••

Fitness evaluation used 11 meaningful and different sources to compute the ViSQOL v3 measure.

Each source is used for 4 different shifting.

C. Again slide 33



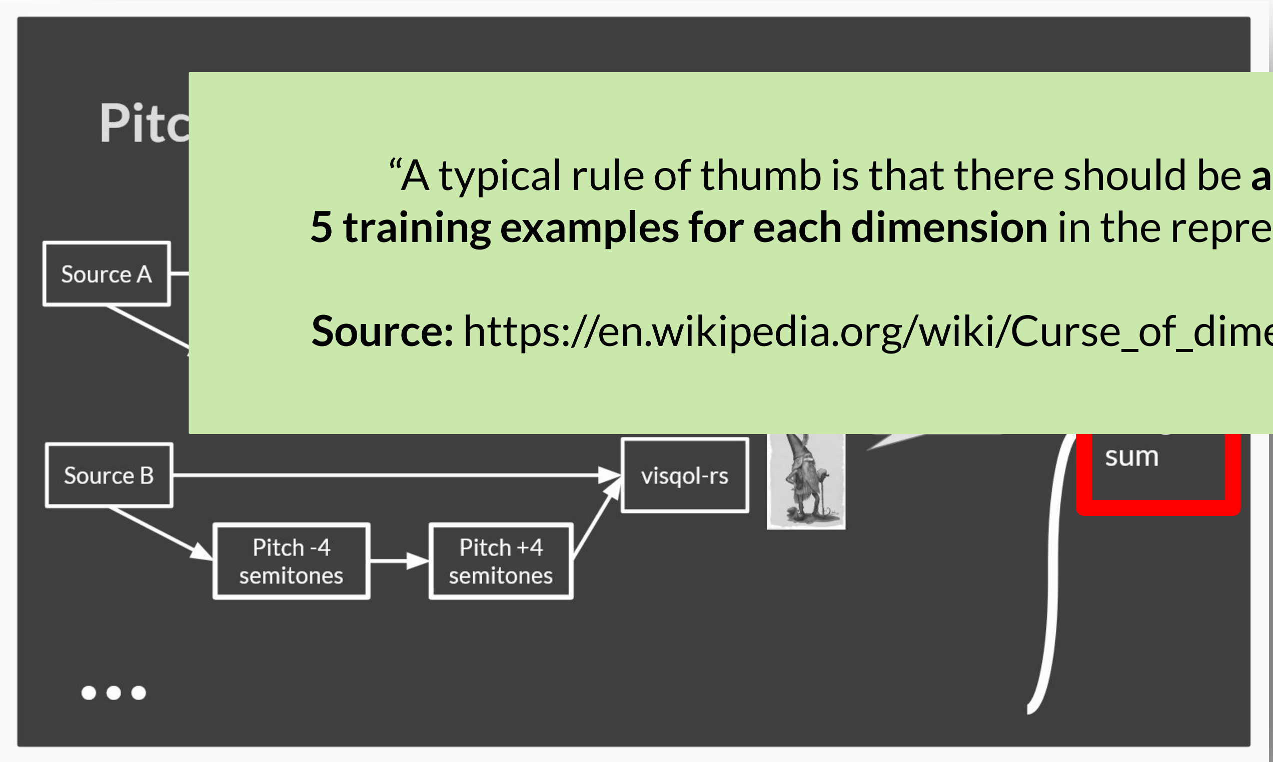
Pitch-shifting objective measure

Fitness evaluation used 11 meaningful and different sources to compute the ViSQOL v3 measure.

Well, that's not enough sources!

Need more data else overfitting.

C. Again slide 33



“A typical rule of thumb is that there should be **at least 5 training examples for each dimension** in the representation.”

Source: https://en.wikipedia.org/wiki/Curse_of_dimensionality

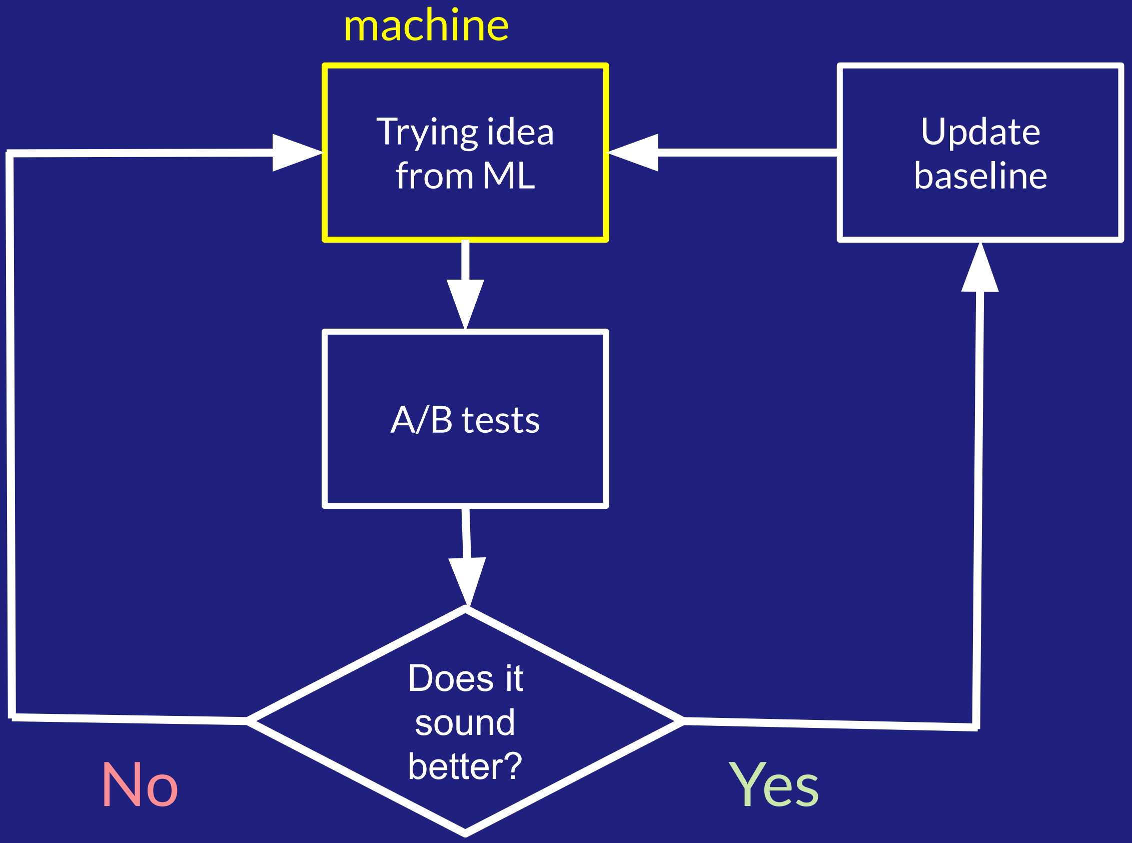
Fitness evaluation
used 11 meaningful
ent
compute
L v3

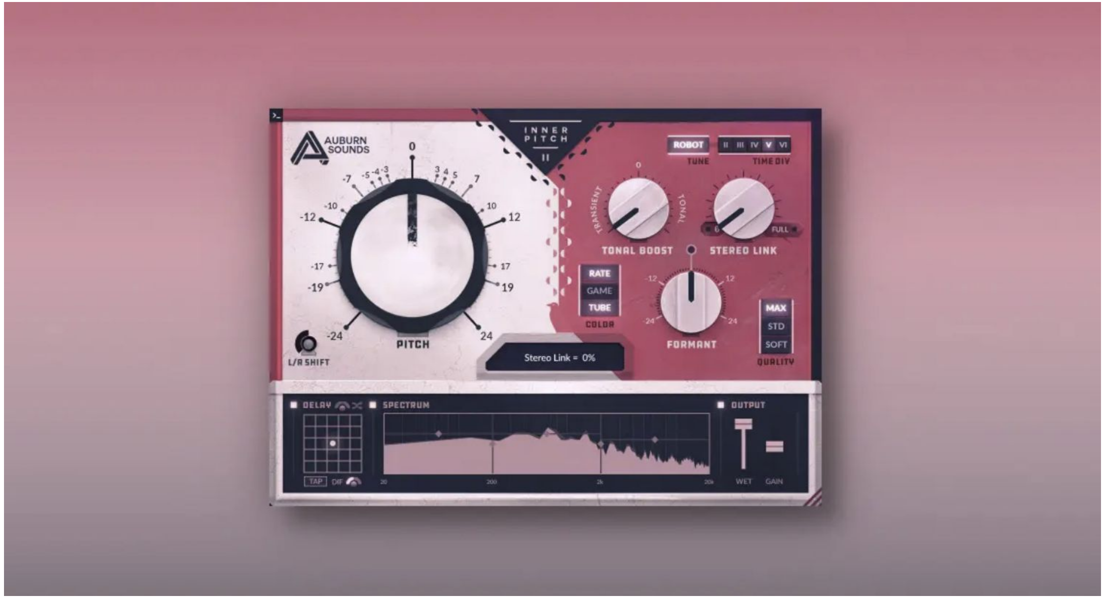
Well, that’s not enough sources!

Need more data else overfitting.

Epilogue

Had to assess each change manually





Stereo Link = 0%

- Shipped in Inner Pitch v2 (Feb 2025)

Tool is on GitHub

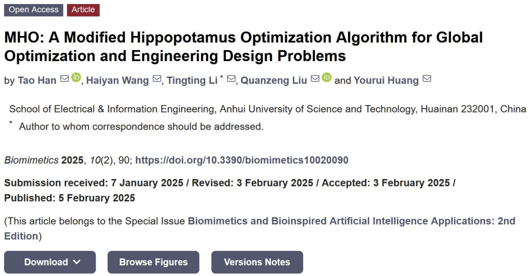
https://github.com/AuburnSounds/Dplug/tools

Questions?

Bonus slide

- Hippopotamus optimization
- Squid Game Optimizer
- Political Optimizer
- Emperor Penguins Colony
- Dujiangyan Irrigation System
- Cuckoo Optimization Algorithm
- Cuckoo Search

All are real meta-heuristic algorithms.



MHO: A Modified Hippopotamus Optimization Algorithm for Global Optimization and Engineering Design Problems

by Tao Han, Haiyan Wang, Tingting Li *, Quanzeng Liu and Yourui Huang

School of Electrical & Information Engineering, Anhui University of Science and Technology, Huainan 232001, China

* Author to whom correspondence should be addressed.

Biomimetics **2025**, *10*(2), 90; https://doi.org/10.3390/biomimetics10020090

Submission received: 7 January 2025 / Revised: 3 February 2025 / Accepted: 3 February 2025 / Published: 5 February 2025

(This article belongs to the Special Issue Biomimetics and Bioinspired Artificial Intelligence Applications: 2nd Edition)

Hippotamus algorithm,
one of the most downloaded
papers of 2024.

Bonus slide

- Hippopotamus optimization
- Squid Game Optimizer
- Political Optimizer
- Emperor Penguins Colony
- Dujiangyan Irrigation System
- Cuckoo Optimization Algorithm
- Cuckoo Search

All are real meta-heuristic algorithms.