Approaches to Live Performance and Composition with Machine Learning and Music Information Retrieval Analysis

Ted Moore

CHIMEFest 2019

My Practice

- Composer (electronics + acoustic instruments)
- Improviser (electronics w/ acoustic collaborators)
- Coder (SuperCollider, Processing, openFrameworks, Python)
- Theatrical Sound Designer

Interest in Music Information Retrieval & Machine Learning

- MIR workshop at CCRMA summer 2018
- In what new ways can I approach sound?
- What can an algorithm do for (with) me? What can it tell me?
- Computational thinking
- What other routes are there to the same goal?

What is the goal?

Make sounds and forms that I find artistically compelling.

Today I share 3 examples of using these tools in that pursuit.

1. Gestural Control in MIR Space

playing random grains from a collection of samples

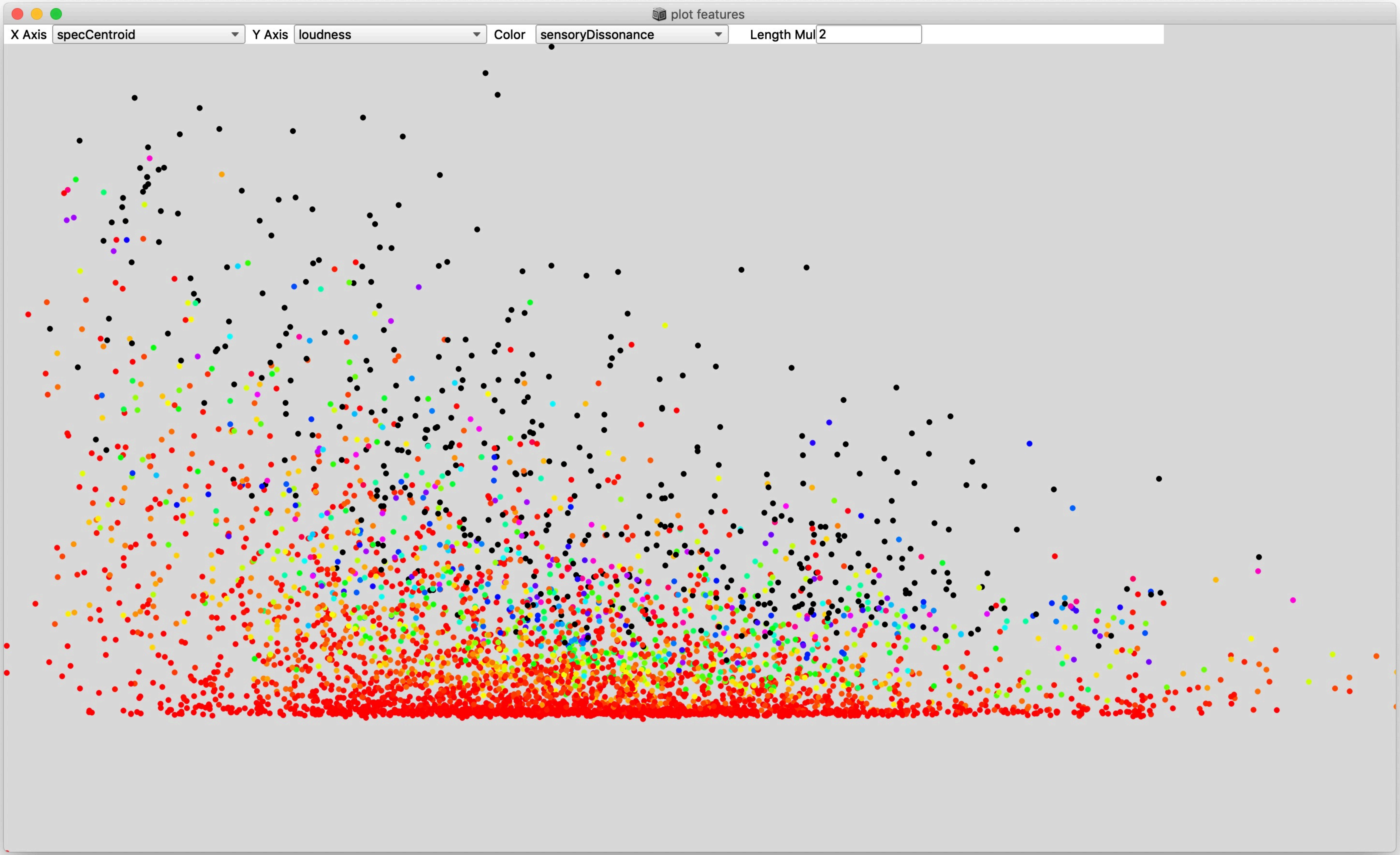
Music Information Retrieval Class in SuperCollider

- 23 Dimensions of Analysis
- Onset Detection
- NRT Analysis of Files and Corpus
- Live Analysis along same Dimensions
- Returns MIRAnalysisFile, an object of its own

0: amplitude
1: fftCrest
2: fftSlope
3: fftSpread
4: loudness
5: sensoryDissonance
6: specCentroid
7: specFlatness
8: specPcile
9: zeroCrossing
10: mfcc01
11: mfcc02
12: mfcc03
13: mfcc04
14: mfcc05
15: mfcc06
16: mfcc07
17: mfcc08
18: mfcc09
19: mfcc10
20: mfcc11
21: mfcc12
22: mfcc13

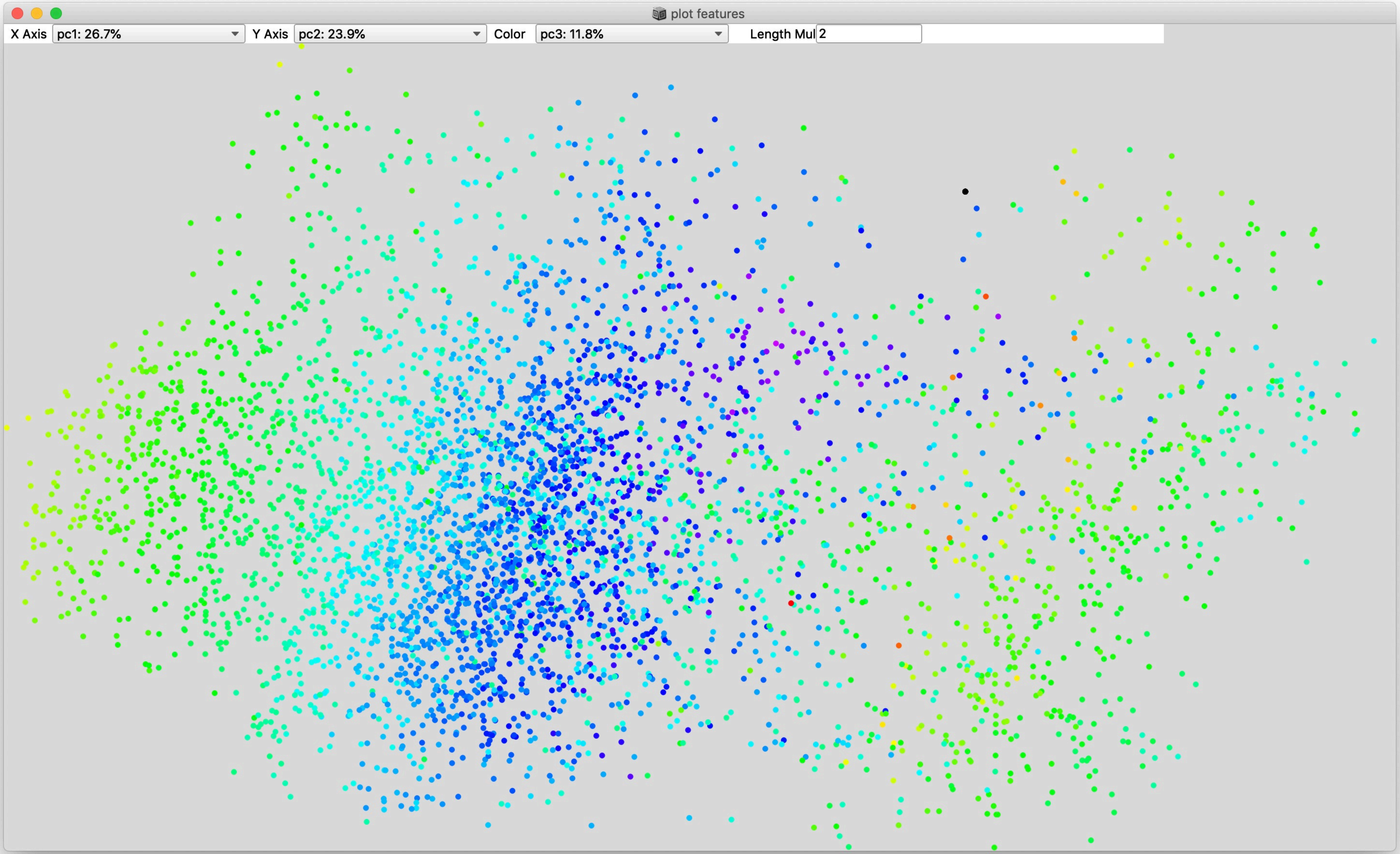
MIRCorpus class in SuperCollider

- MIRAnalysis on source corpus
- NRT
- Returns MIRCorpusItem


plot features
X Axis
specCentroid
Y Axis
loudness
Color
sensoryDissonance
Length Mul
2

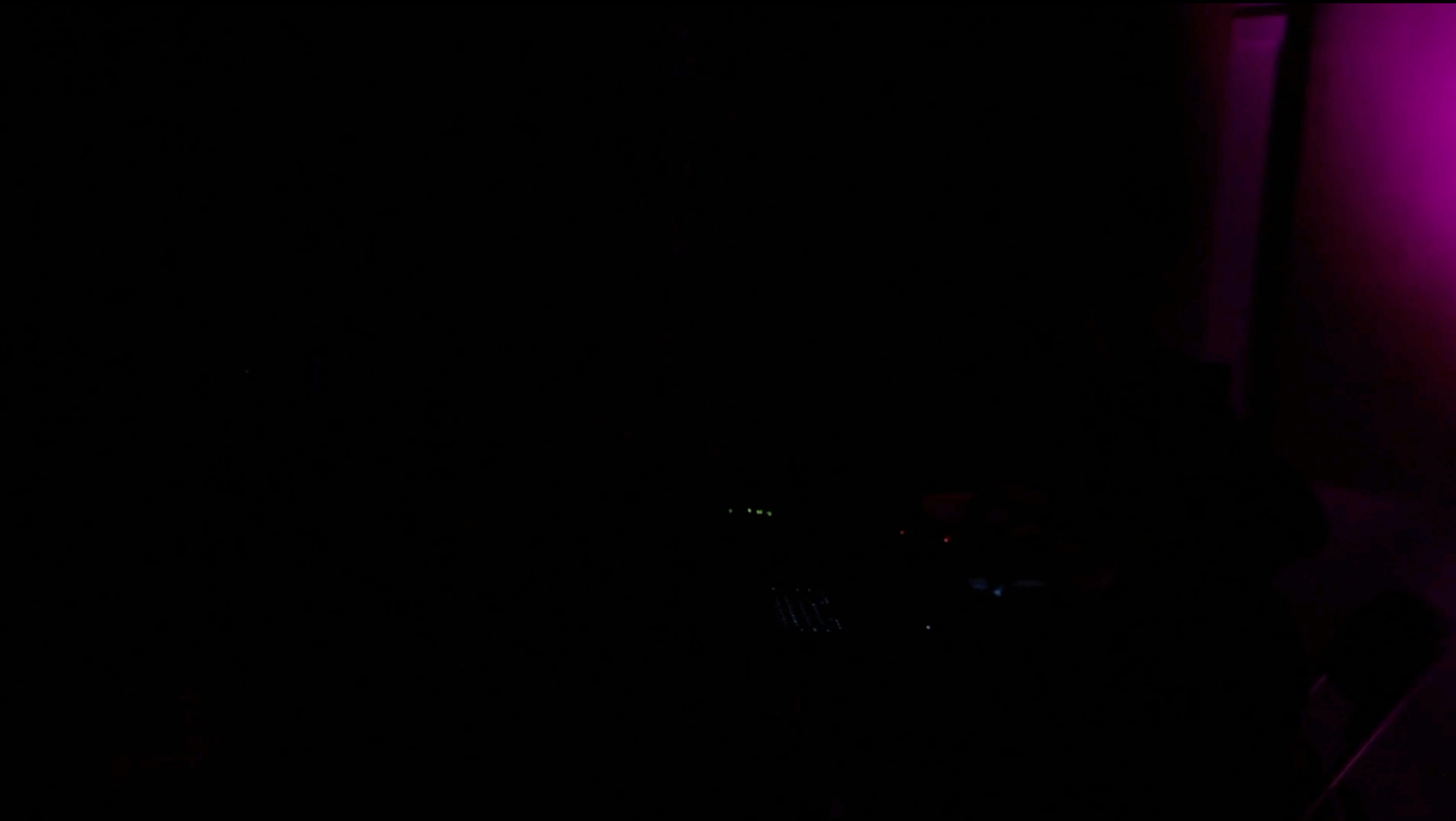
Principal Component Analysis

- Reduce the number of dimensions in a data set
- Maintain the variance in the data set
- Remove redundancy

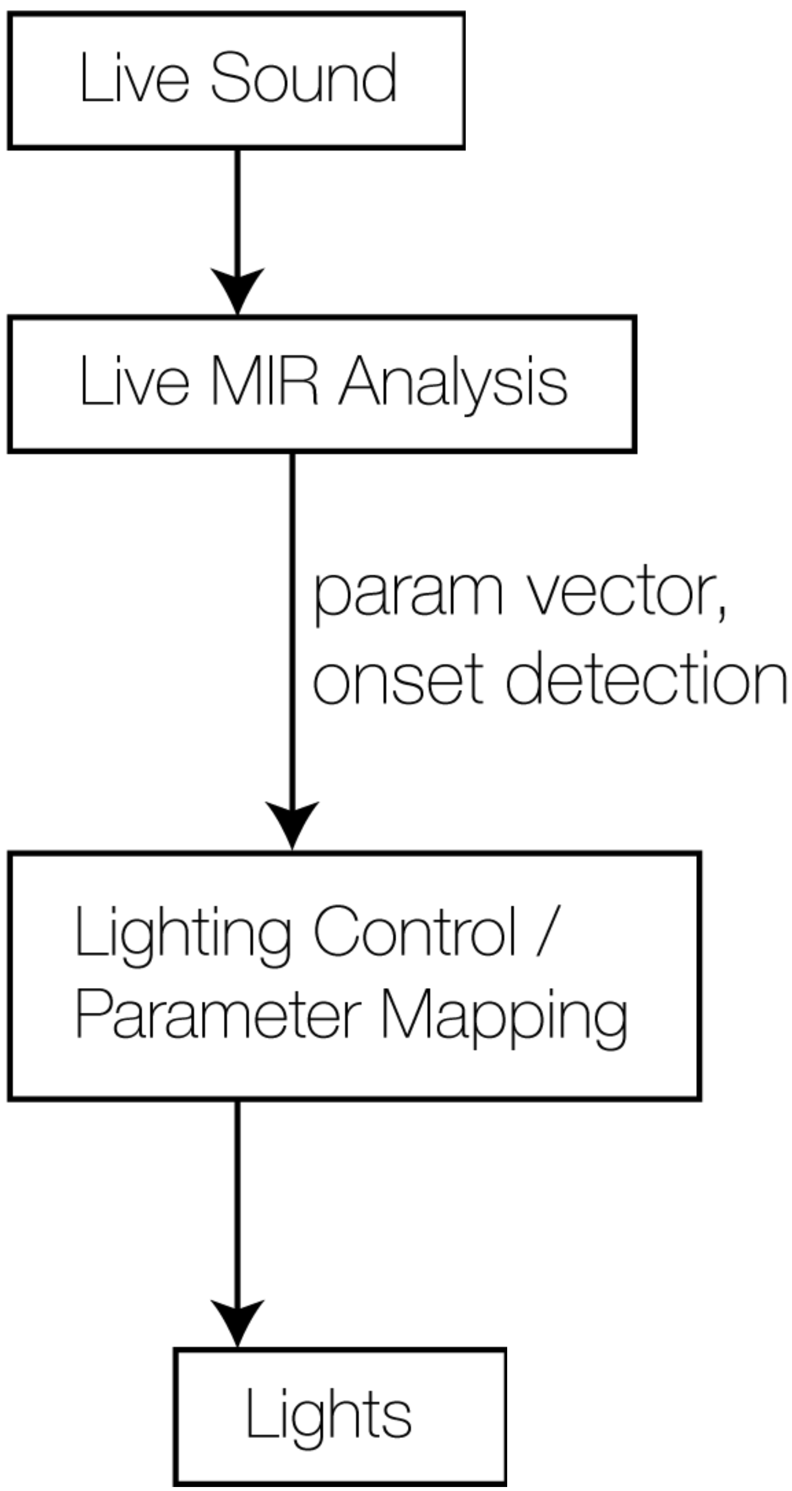

plot features
X Axis
pc1: 26.7%
Y Axis
pc2: 23.9%
Color
pc3: 11.8%
Length Mul
2

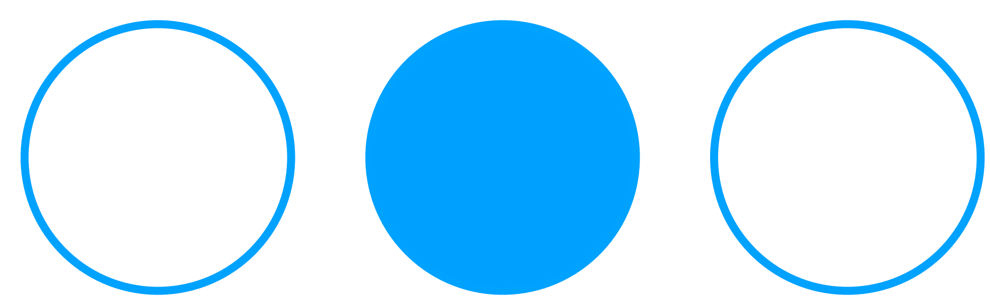
demo time

2. Live Sound Classification

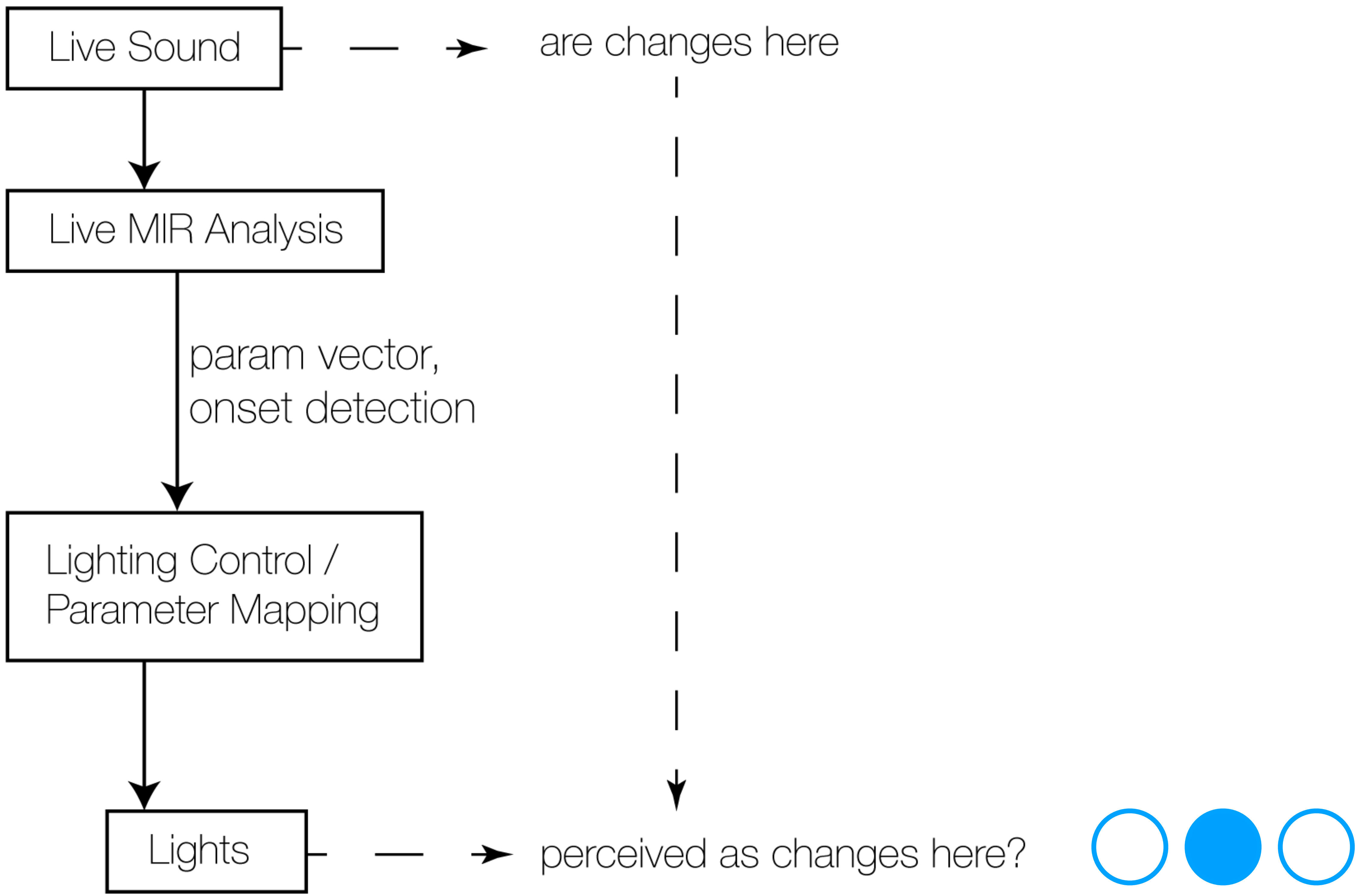


Machine Listening System

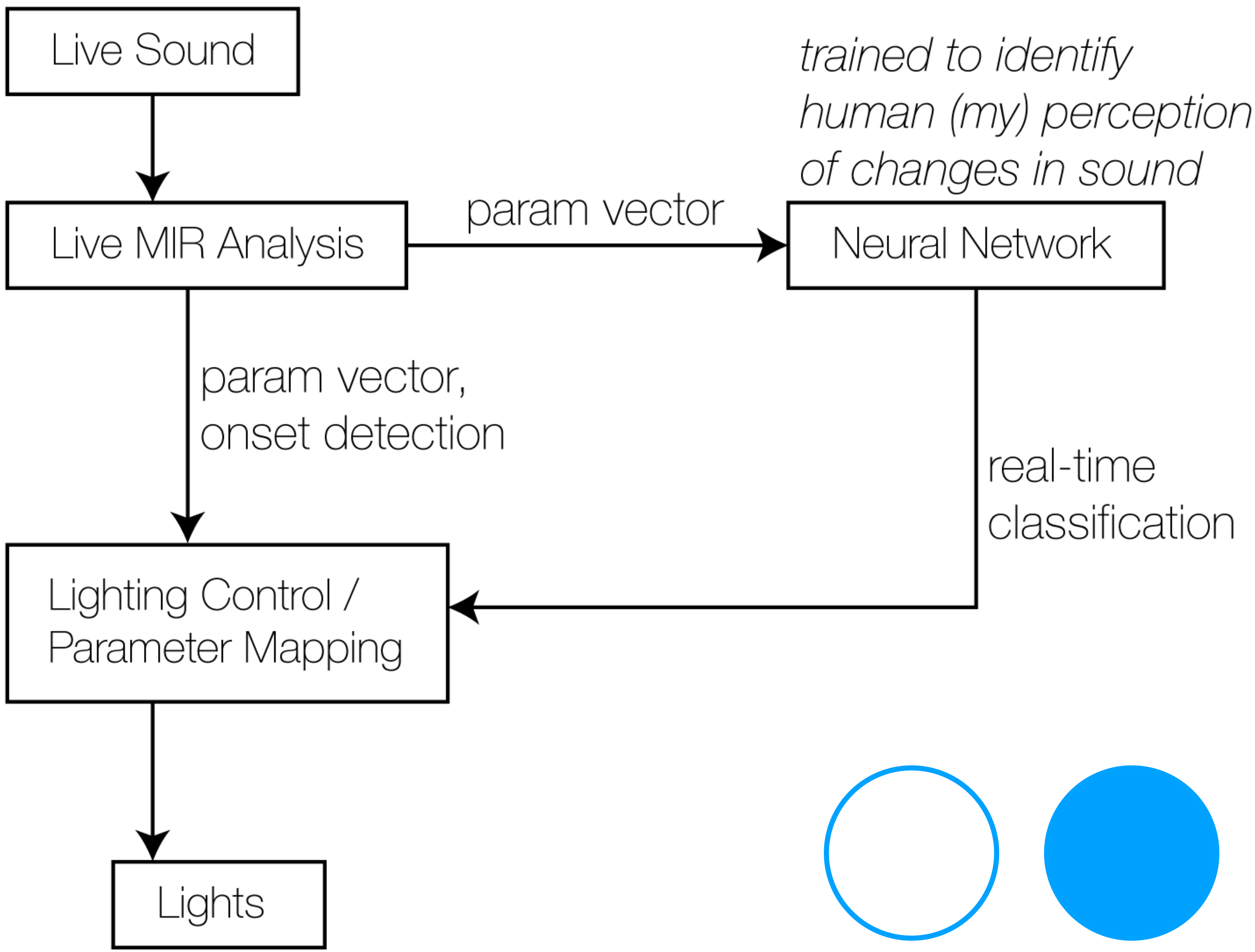




Machine Listening System



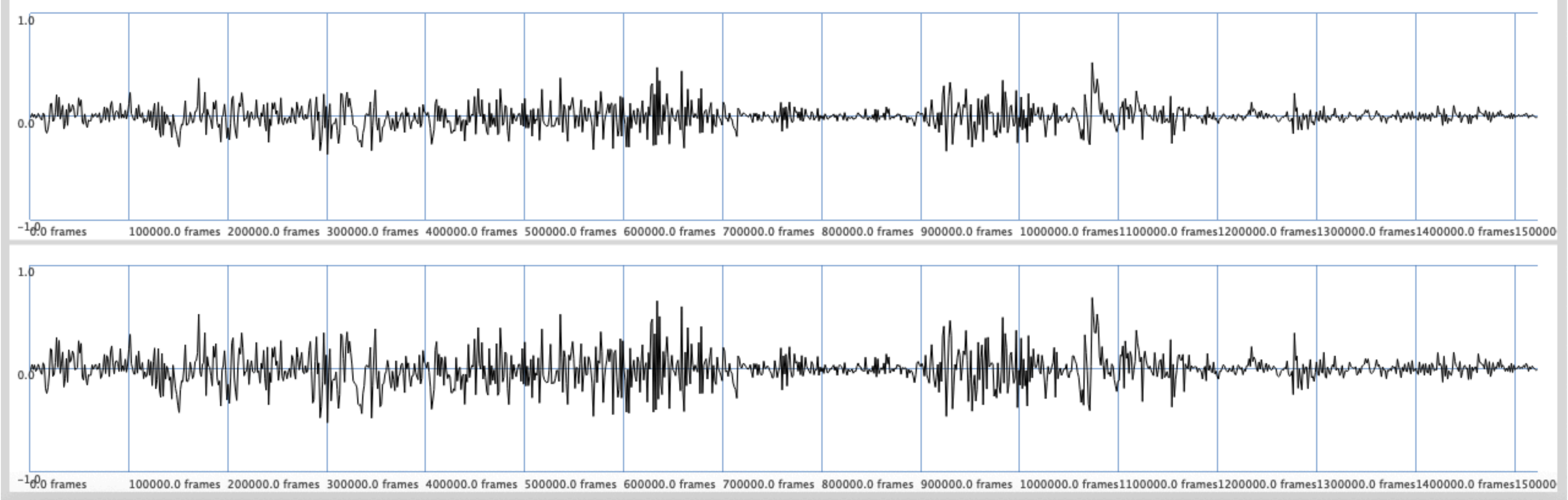
Machine Learning System

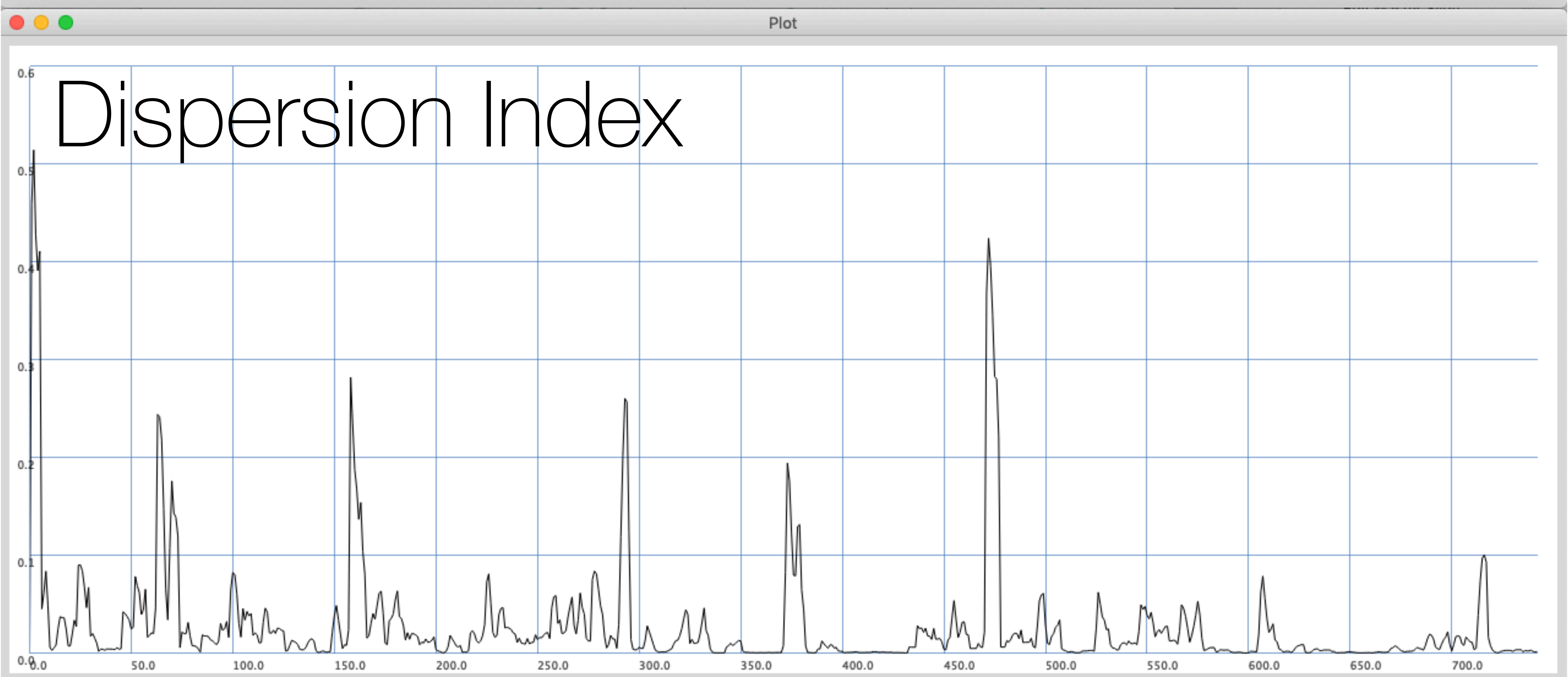


distorted_noise
high_squeal
low_impulses
sus_noise_quiet

```
NeuralNetwork {
    var <>net, <>learningRate, e = 2.71828, <shape, <>activation, <>normalizedRanges;

    *new {
        arg shape,learningRate = 0.05,activation = "relu",normalizedRanges;
        ^super.new.init(shape,learningRate,activation,normalizedRanges);
    }

    init {
        arg shape_,learningRate_ = 0.05,activation_ = "relu",normalizedRanges_;
        shape = shape_;
        activation = activation_;
        learningRate = learningRate_;
        normalizedRanges = normalizedRanges_;

        net = shape.collect({
            arg nNeurons, i;
            var data = (
                vals:Array.fill(nNeurons,{0}),
            );
            if(i > 0,{
                // not input layer;
                data.biases = Array.fill(nNeurons,{rrand(-1.0,1.0)});
                data.weights = Array.fill(shape[i],{
                    Array.fill(shape[i-1],{rrand(-1.0,1.0)});
                });
            });
            data;
        });
```

```
feedLights = FeedLightMaster([

    // distorted noise
    FeedLightMode(nLights,[
        FeedLightGroup([
            \amplitude,\myAmp,\v,ControlSpec(0.01,1,\exp),
            \specCentroid,ControlSpec(50,5000,\exp),\h,ControlSpec(0.5,0.7),
            \specFlatness,nil.asSpec,\s,ControlSpec(1,0.3)
        ]),
        FeedLightGroup([
            \amplitude,\myAmp,\v,ControlSpec(0.01,1,\exp),
            \specCentroid,ControlSpec(50,5000,\exp),\h,ControlSpec(0.4,0.6),
            \specFlatness,nil.asSpec,\s,ControlSpec(1,0.3)
        ])
    ]),

    // high squeal
    FeedLightMode(nLights,[
        FeedLightGroup([
            \amplitude,\myAmp,\s,ControlSpec(0.5,0.9),
            \zeroCrossing,ControlSpec(3000,10000,\exp),\h,ControlSpec(0,0.25),
            \constant,1,\v,nil
            //\specFlatness,nil.asSpec,\w,ControlSpec(0,255),
            //\zeroCrossing,ControlSpec(50,6000,\exp),\r,ControlSpec(0,255)
        ]),
        FeedLightGroup([
            \amplitude,\myAmp,\s,ControlSpec(0.5,0.9)
```


1.0
0.0
-1.0
0.0 frames
100000.0 frames
200000.0 frames
300000.0 frames
400000.0 frames
500000.0 frames
600000.0 frames
700000.0 frames
800000.0 frames
900000.0 frames
1000000.0 frames
1100000.0 frames
1200000.0 frames
1300000.0 frames
1400000.0 frames


Plot
Dispersion Index
0.6
0.5
0.4
0.3
0.2
0.1
0.0
50.0
100.0
150.0
200.0
250.0
300.0
350.0
400.0
450.0
500.0
550.0
600.0
650.0
700.0

3. Corpus Concatenation

ConcatSynthNRT class in SuperCollider

Render in many ways, flexibility for rendering from different types of data

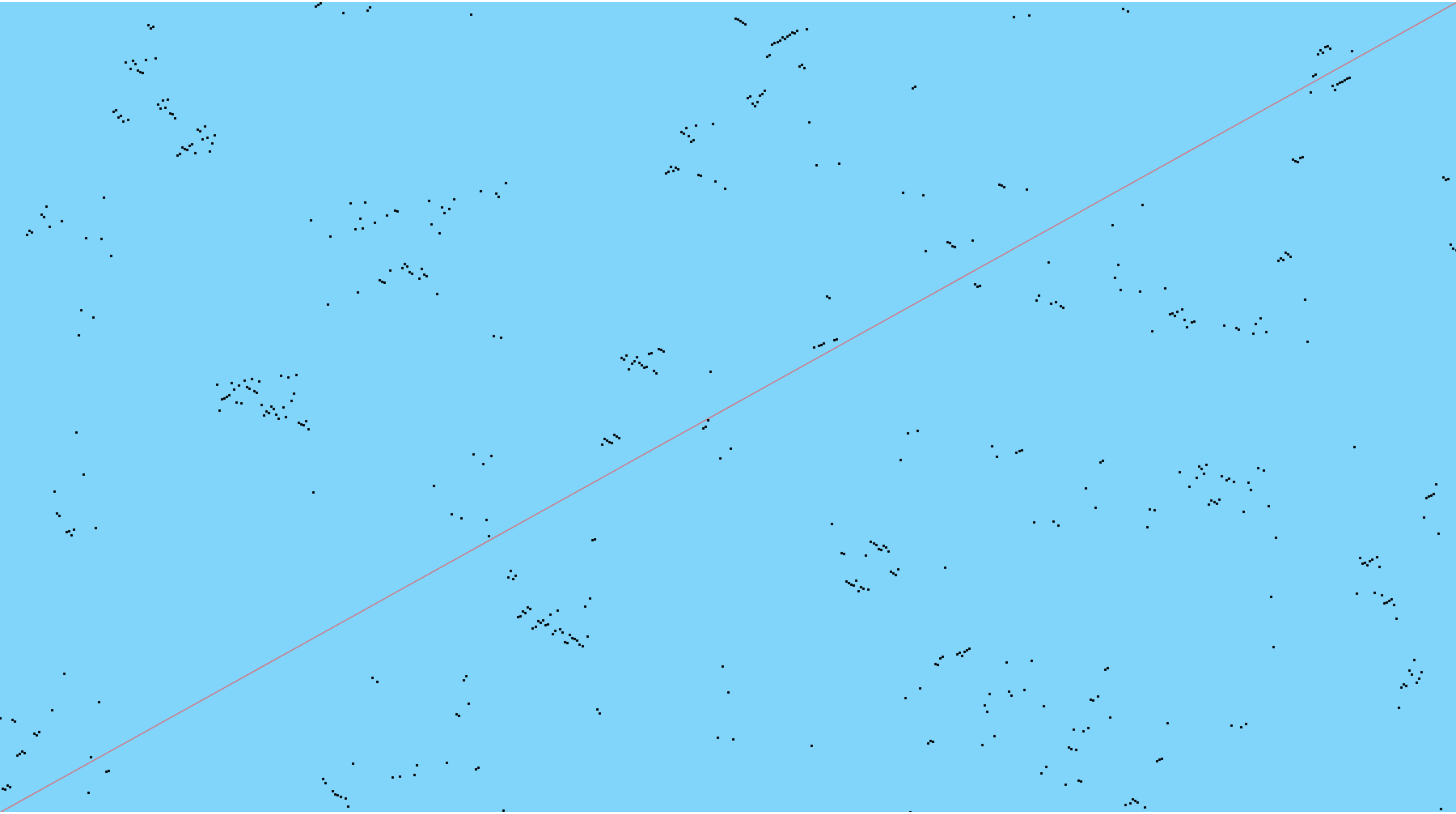
- renderFromCorpusAndFilePath()
  - kNN
- renderFromCorpusAndRawFrames()
  - kNN
- renderFromArrayOfCorpusItems()
  - from given path

what does the fox say?

what does the fox say?

pathfinding through MIR space: empowering algorithms to organize time

what does the sax say?



what does the corpus say?

reiny_bells 01

noisy_laptop 01

saxophone 01

eurorack 01

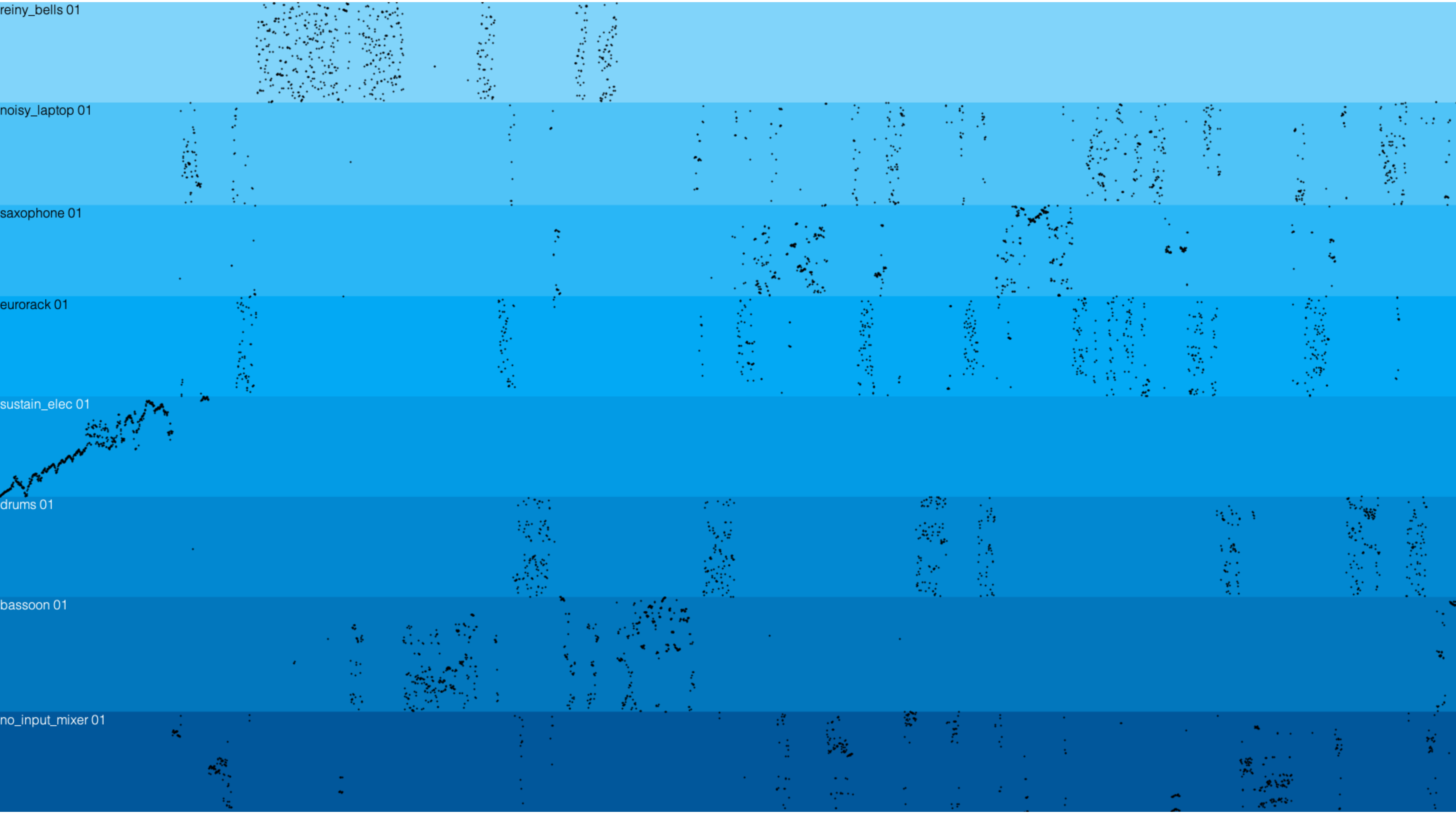
sustain_elec 01

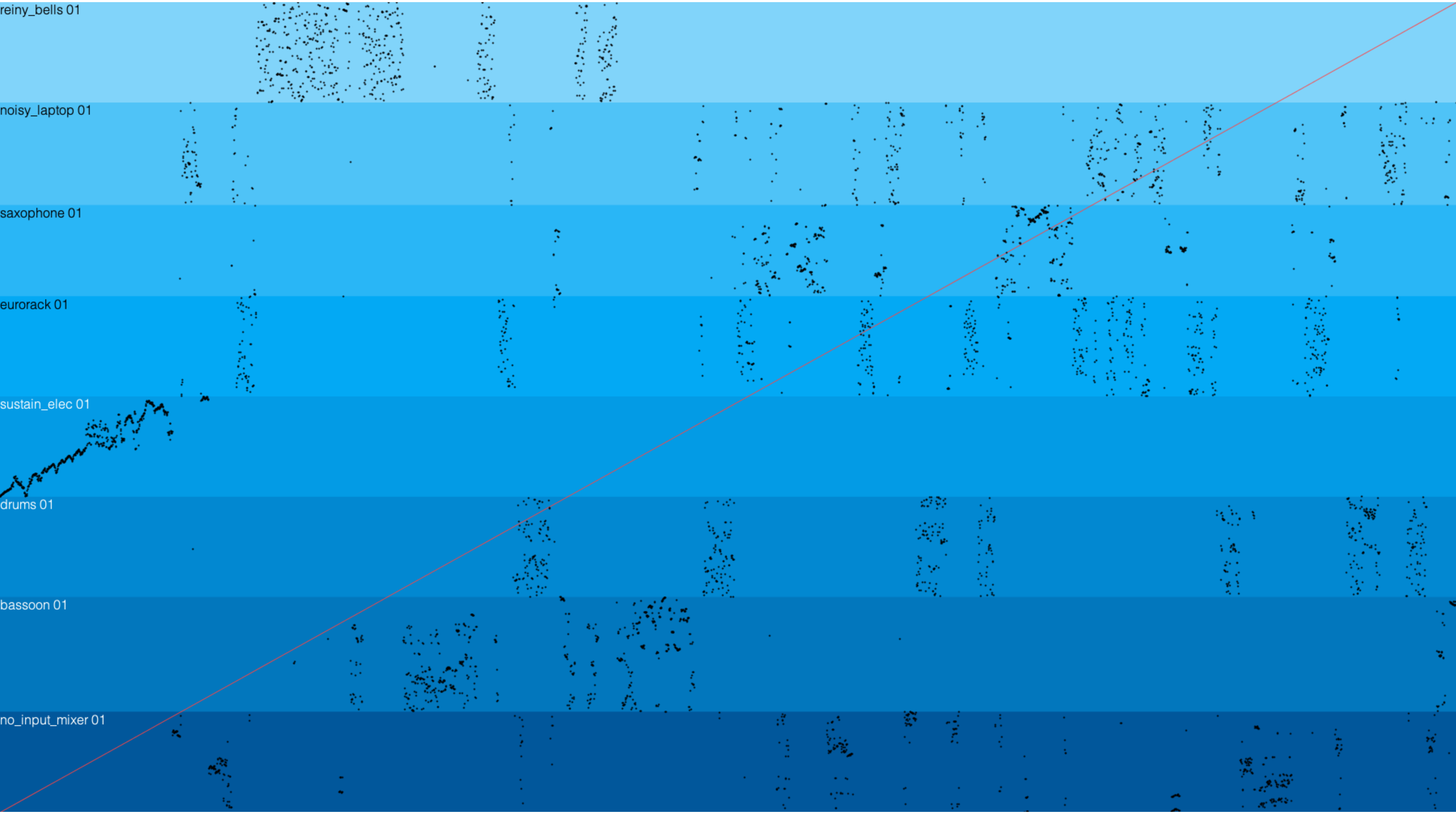
drums 01

bassoon 01

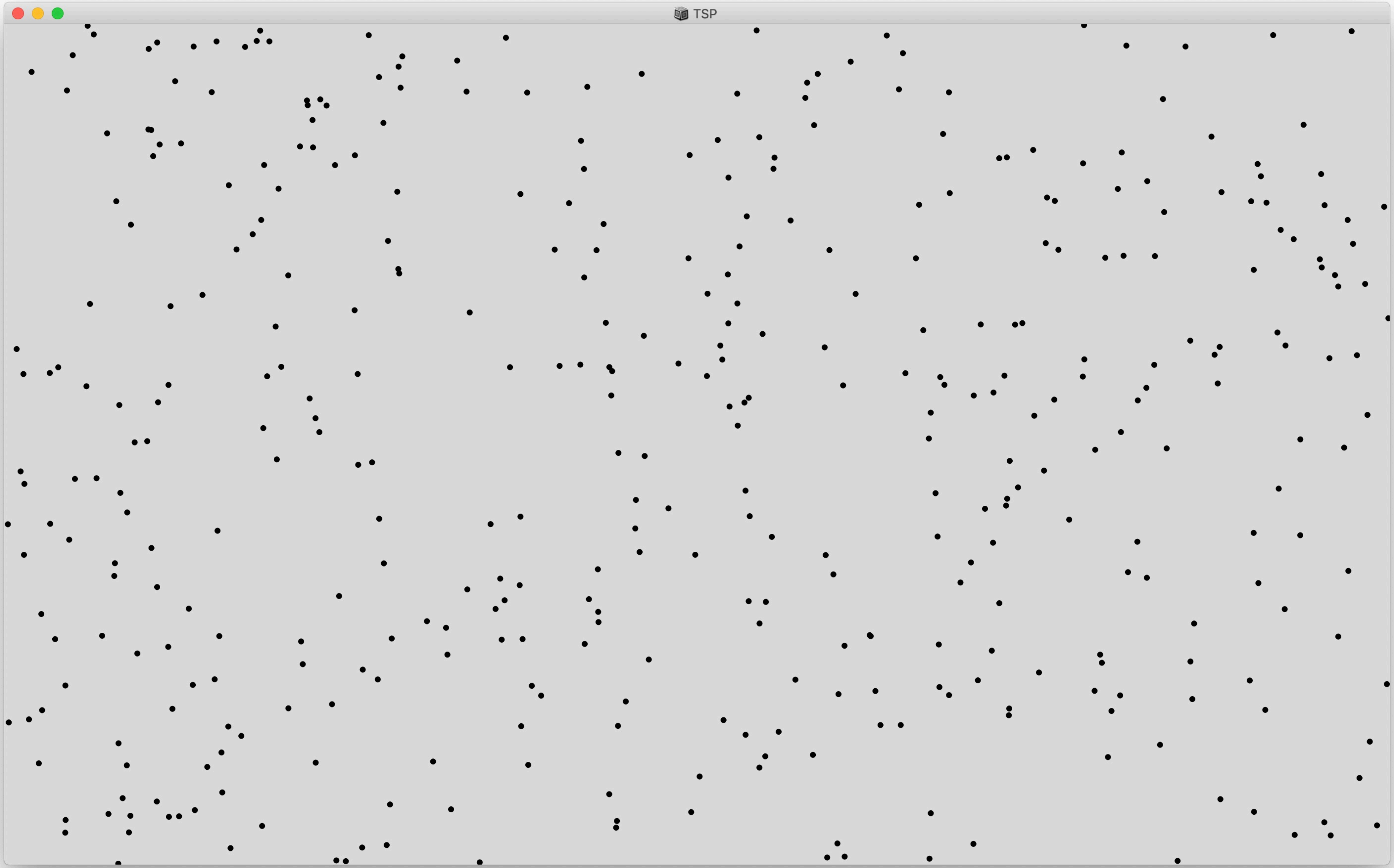
no_input_mixer 01

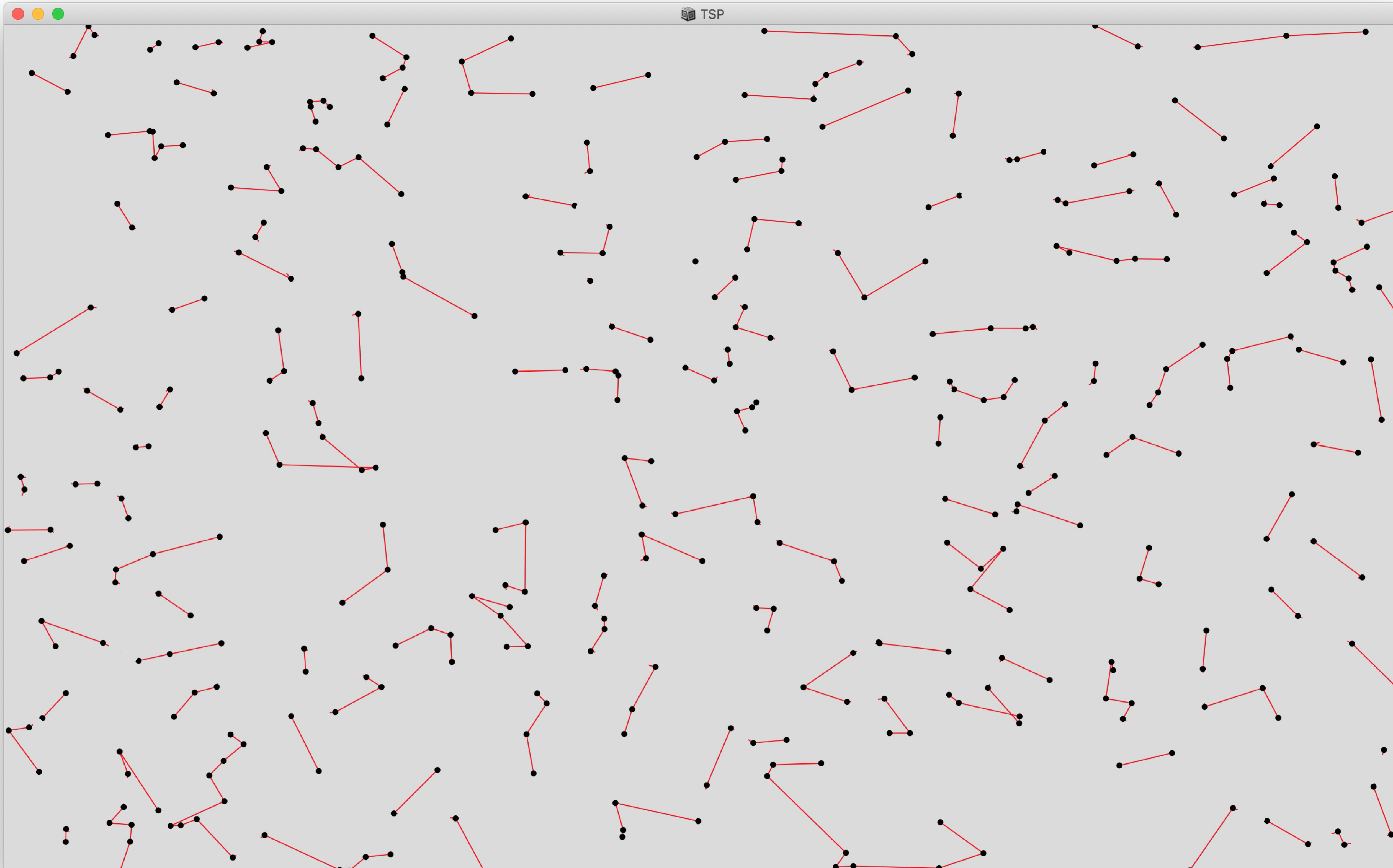

reiny_bells 01
noisy_laptop 01
saxophone 01
eurorack 01
sustain_elec 01
drums 01
bassoon 01
no_input_mixer 01

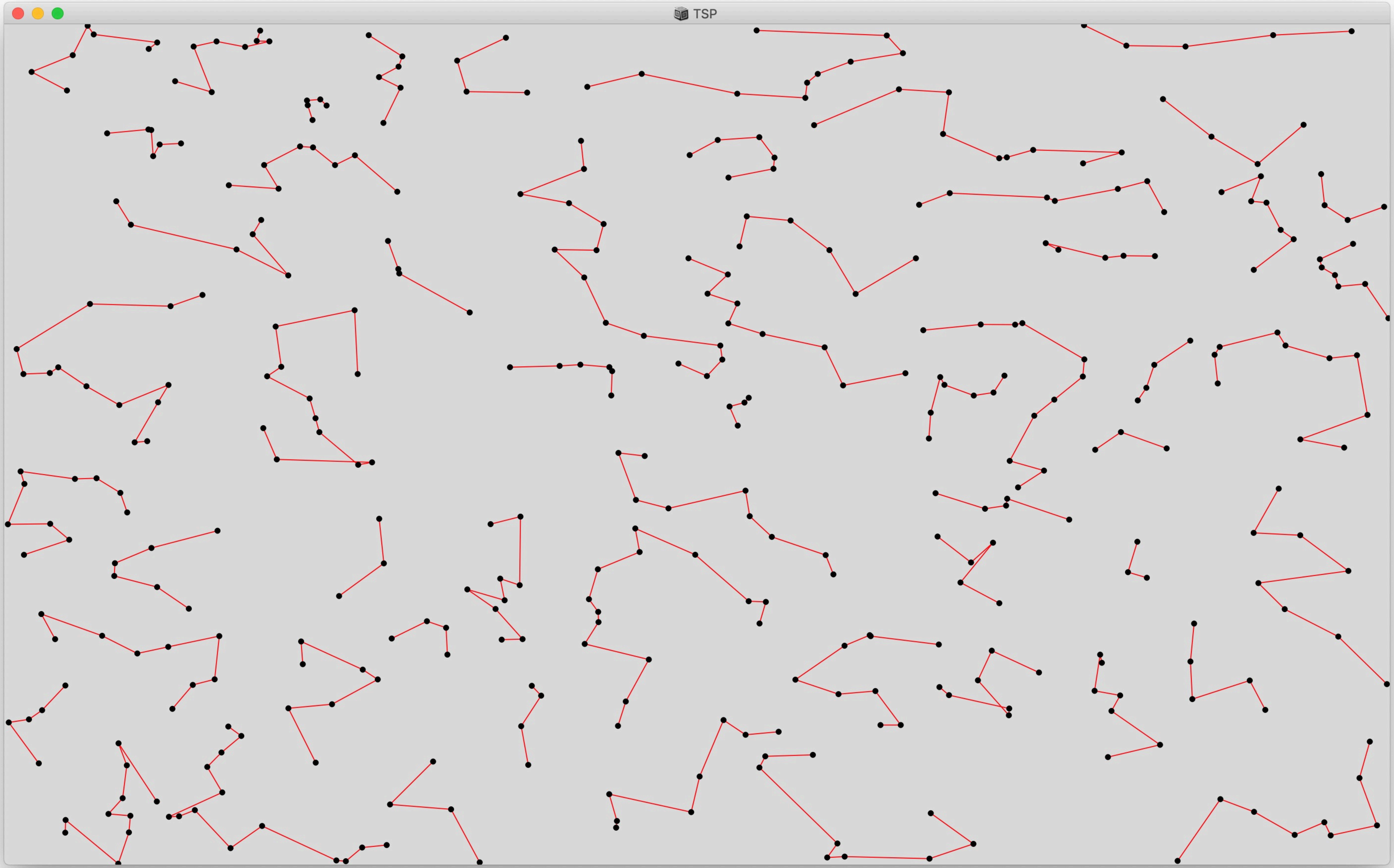

reiny_bells 01
noisy_laptop 01
saxophone 01
eurorack 01
sustain_elec 01
drums 01
bassoon 01
no_input_mixer 01

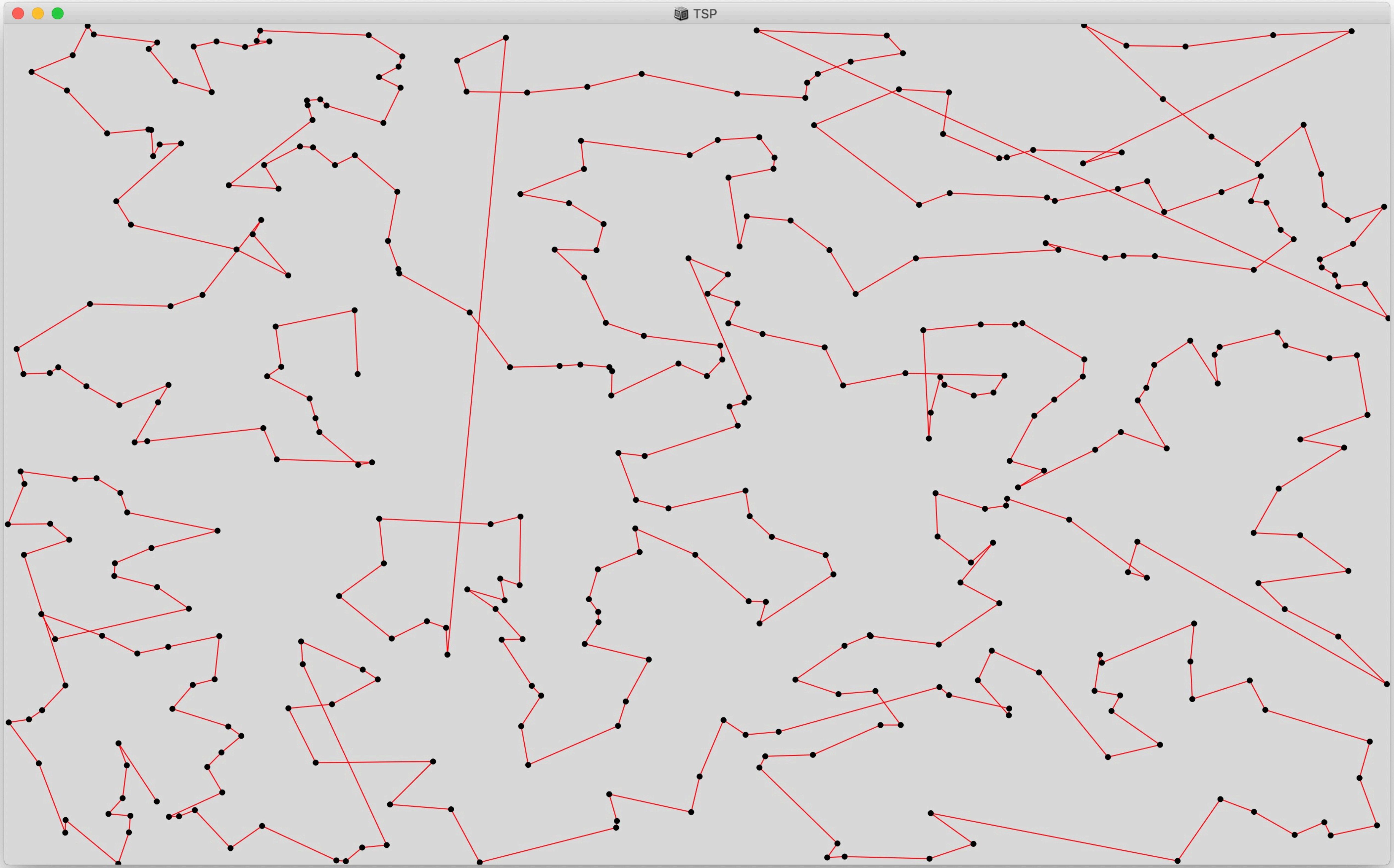

reiny_bells 01
noisy_laptop 01
saxophone 01
eurorack 01
sustain_elec 01
drums 01
bassoon 01
no_input_mixer 01

machine composed timbral fusion, gesture, & form


TSP


TSP


TSP


TSP

pathfinding through FFT space: towards machine learning “synthesis”

```
FFTNRT {

    *fft {
        arg filePath,  action, fftSize = 2048, overlap = 2;
        SoundFile.use(filePath,{
            arg sf;
            var data, window, frames, currentSample = 0,hopSamples, fft, imag;

            hopSamples = fftSize / overlap;

            data = FloatArray.newClear(sf.numFrames * sf.numChannels);
            sf.readData(data);

            // sum to mono
            // TODO: process stereo
            if(sf.numChannels > 1,{
                data = sf.numFrames.collect({
                    arg frameI;
                    var val = 0;
                    sf.numChannels.do({
                        arg chanI;
```

pathfinding through FFT space: towards machine learning “synthesis”

```
Python running for 08:53:33.471
Python running for 08:53:34.471
Python running for 08:53:35.471
[ 6982, 6981, 4968, 8067, 8002, 800
```

saxophone, no input mixer, laptop, eurorack

“The view according to which the novelty of a work guarantees its quality is often expressed in electroacoustic music circles, and for some it is the only criterion of worthiness.”

–Francis Dhomont, *For classicism*

Future Directions

- Introduce more sequence (time) based algorithms (RNNs, HMMs), have them “help” in real-time
- Understand lower dimensional space using VAE (instead of PCA)
- Machine Learning Synthesis (composing sequential FFT frames, GANs)
- Concatenative Synthesis not based on kNN, but on user defined Neural Network mappings
- Live Concatenative Synthesis
- Path find through multidimensional space of laptop improvisation interface (track MIDI & OSC data, MIR data)
- tedbot

Thank you. Questions?