Carnatic Music Synthesis

in Clojure

@SrihariSriraman

@nilenso

What this talk is about

- Modeling Carnatic Music in code
- Making a machine sing Carnatic Music
- Synthesis, not generation

Should we listen to some

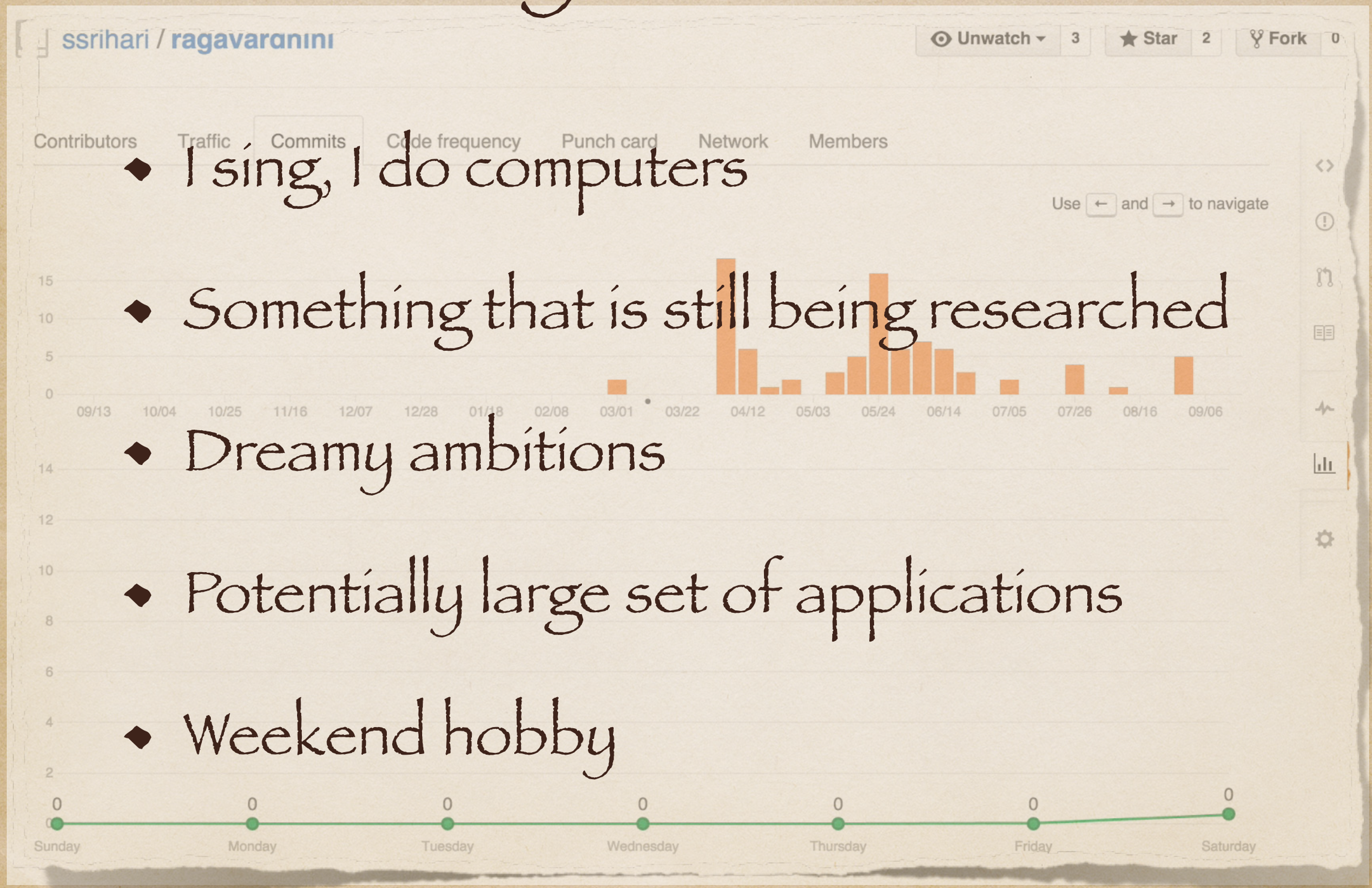
right now?

if not, we'll keep the suspense for later

What is Carnatic Music?

- South Indian Classical Music
- Gamakams
- Mostly a vocal tradition
- Rich in compositions; revolves around it
- Extempore / Manodharma

Why I do this



- I sing, I do computers
- Something that is still being researched
- Dreamy ambitions
- Potentially large set of applications
- Weekend hobby

What I'll cover

- Shruthi, Swarams and, Ragams

  in music, and in Clojure

- A demo of the basics

  play some prescriptive notation in a given raga in the REPL

- Gamakams – modeling, and rendering

  with a demo of course

- The way of the future

  as I see it

Shruthi, Swarams, & Ragams

in Music, and in Clojure

```clojure
(def shruthis
  {:.a  57 :.a# 58 :.b  59 :c   60 :c#  61 :db  61 :d   62
   :d#  63 :eb  63 :e   64 :f   65 :f#  66 :gb  66 :g   67
   :g#  68 :ab  68 :a   69 :a#  70 :bb  70 :b   71 :c.  72})

(def madhya-sthayi-sthanams
  {:s 0
   :r1 1 :r2 2 :r3 3
   :g1 2 :g2 3 :g3 4
   :m1 5 :m2 6
   :p 7
   :d1 8 :d2 9
   :d3 10
   :n1 9 :n2 10 :n3 11})

(def simple-swarams
  [:s :r :g :m :p :d :n])

(def sthayis
  {:anumandra {:position :before :dots ".." :difference -24}
   :mandra    {:position :before :dots "."  :difference -12}
   :madhya    {:position :none   :dots ""   :difference 0}
   :thara     {:position :after  :dots "."  :difference 12}
   :athithara {:position :after  :dots ".." :difference 24}})
```

Ragam

```
1
2 {{:name :suryakantam,
3   :num 17,
4   :arohanam [:s :r1 :g3 :m1 :p :d2 :n3 :s.],
5   :avarohanam (:s. :n3 :d2 :p :m1 :g3 :r1 :s)}
6  {:sahuli
7   {:arohanam (:s :g3 :m1 :p :n3 :s.),
8    :avarohanam (:s. :n3 :d2 :p :m1 :g3 :r1 :s)},
9   :shuddhalalita
10  {:arohanam (:s :r1 :g3 :m1 :d2 :n3 :s.),
11   :avarohanam (:s. :n3 :d2 :m1 :g3 :r1 :s)},
12  :shuddhagandharvam
13  {:arohanam (:s :r1 :g3 :m1 :p :d2 :n3 :s.),
14   :avarohanam (:s. :d2 :p :m1 :r1 :s)},
15  :kanakacala
16  {:arohanam (:s :g3 :p :n3 :s.), :avarohanam (:s. :d2 :m1 :r1 :s)},
17  :kanthiravam
18  {:arohanam (:s :r1 :g3 :p :d2 :n3 :s.),
19   :avarohanam (:s. :n3 :d2 :p :m1 :g3 :r1 :s)},
20
```

```
17545   :deshamukhari
17546   {:arohanam (:s :r1 :g2 :m2 :p :d3 :n3 :d3 :s.),
17547    :avarohanam (:s. :n3 :d3 :n3 :p :m2 :g2 :r1 :s)},
17548   :amrtapancamam
17549   {:arohanam (:s :r1 :g2 :m2 :p :d3 :n3 :s.),
17550    :avarohanam (:s. :n3 :d3 :m2 :g2 :s)},
17551   :rudragandhari
17552   {:arohanam (:s :r1 :g2 :m2 :n3 :s.),
17553    :avarohanam (:s. :n3 :d3 :n3 :s :n3 :p :m2 :r1 :s)},
17554   :jeevantikaa
17555   {:arohanam (:s :m2 :p :d3 :n3 :s.),
17556    :avarohanam (:s. :n3 :p :m2 :g2 :s)},
17557   :samabarbari
17558   {:arohanam (:s :r1 :g2 :n3 :d3 :n3 :s.),
17559    :avarohanam (:s. :n3 :d3 :m2 :g2 :r1 :s)},
17560   :amrapancamam
17561   {:arohanam (:s :r1 :g2 :m2 :p :d3 :n3 :s.),
17562    :avarohanam (:s. :n3 :d3 :m2 :g2 :s)},
17563   :deshavali
17564   {:arohanam (:s :r1 :g2 :m2 :d3 :n3 :d3 :s.),
17565    :avarohanam (:s. :n3 :d3 :m2 :g2 :r1 :s)},
17566   :amapancamam
17567   {:arohanam (:s :r1 :g2 :m2 :p :d3 :n3 :s.),
17568    :avarohanam (:s. :n3 :d3 :m2 :g2 :s)}}}
17569
```

Laya concepts

```
(def kaala-pramanam 80)

(def jathis
  {:chaturasra 4
   :thisra     3
   :khanda     5
   :misra      7
   :sankeerna  9})

(def kalams
  {:vilamba     1
   :chauka      2
   :madhyama    4
   :durita      8
   :ati-durita 16})
```

A Demo of the basics

-> to the REPL

A Demo of the basics

```
1 ragavardhini.demo>
```

Tools and libraries

- SuperCollider synthesis server
- Overtone instrument
- Leipzig melodies
- Postgres fuzzy search

So far...

- Play the scale of a raga
- Fuzzy find a Raga
- Play a phrase
- Play a phrase in a given raga
- Play some prescriptive notation

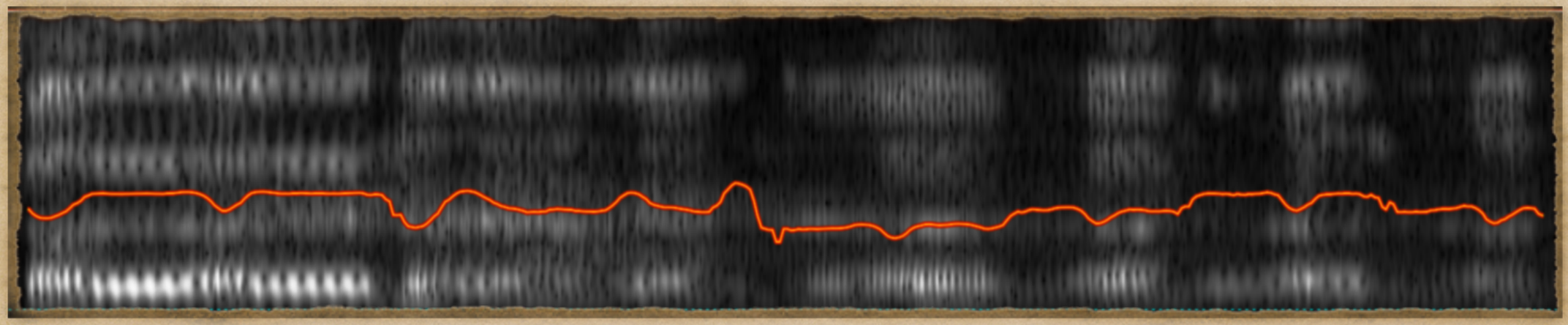
But..

..that doesn't sound like

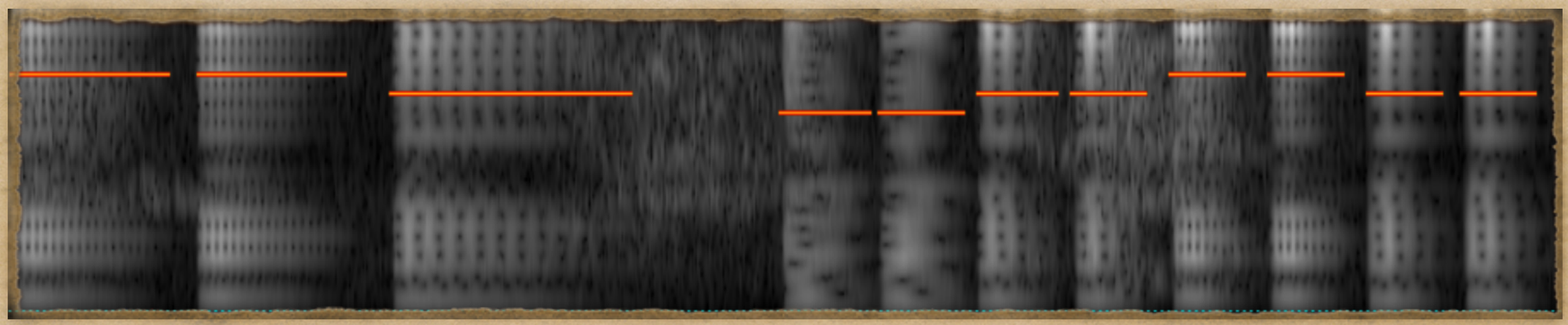
Carnatic Music, does it?

Enter pitch graphs

Me

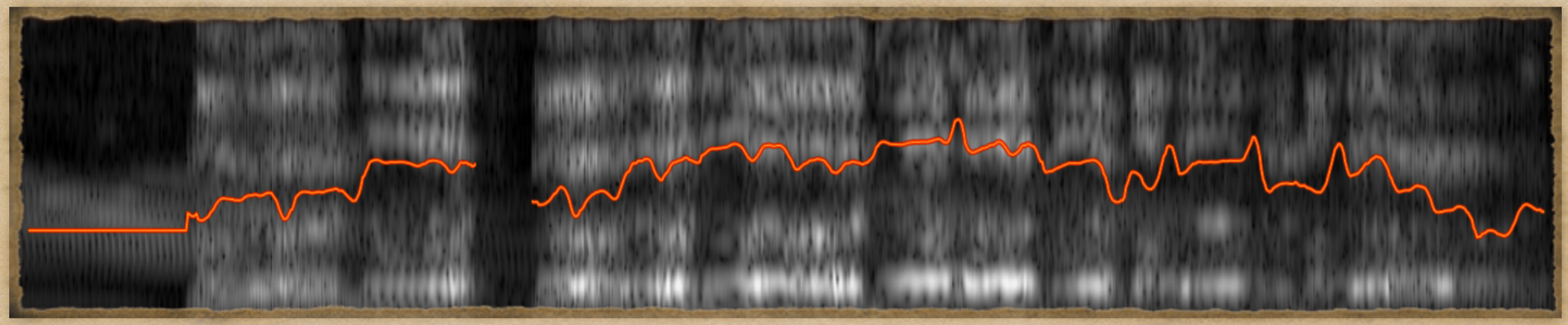


Machine

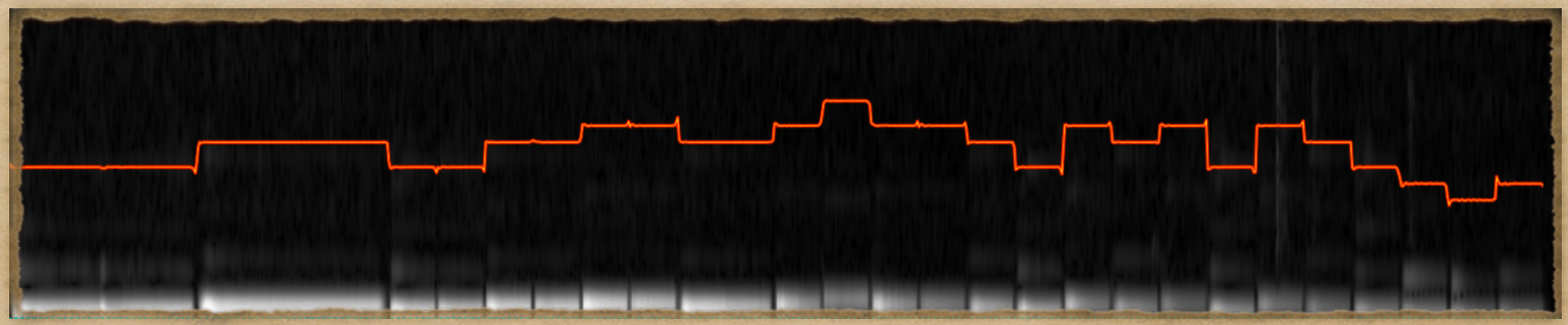


another phrase

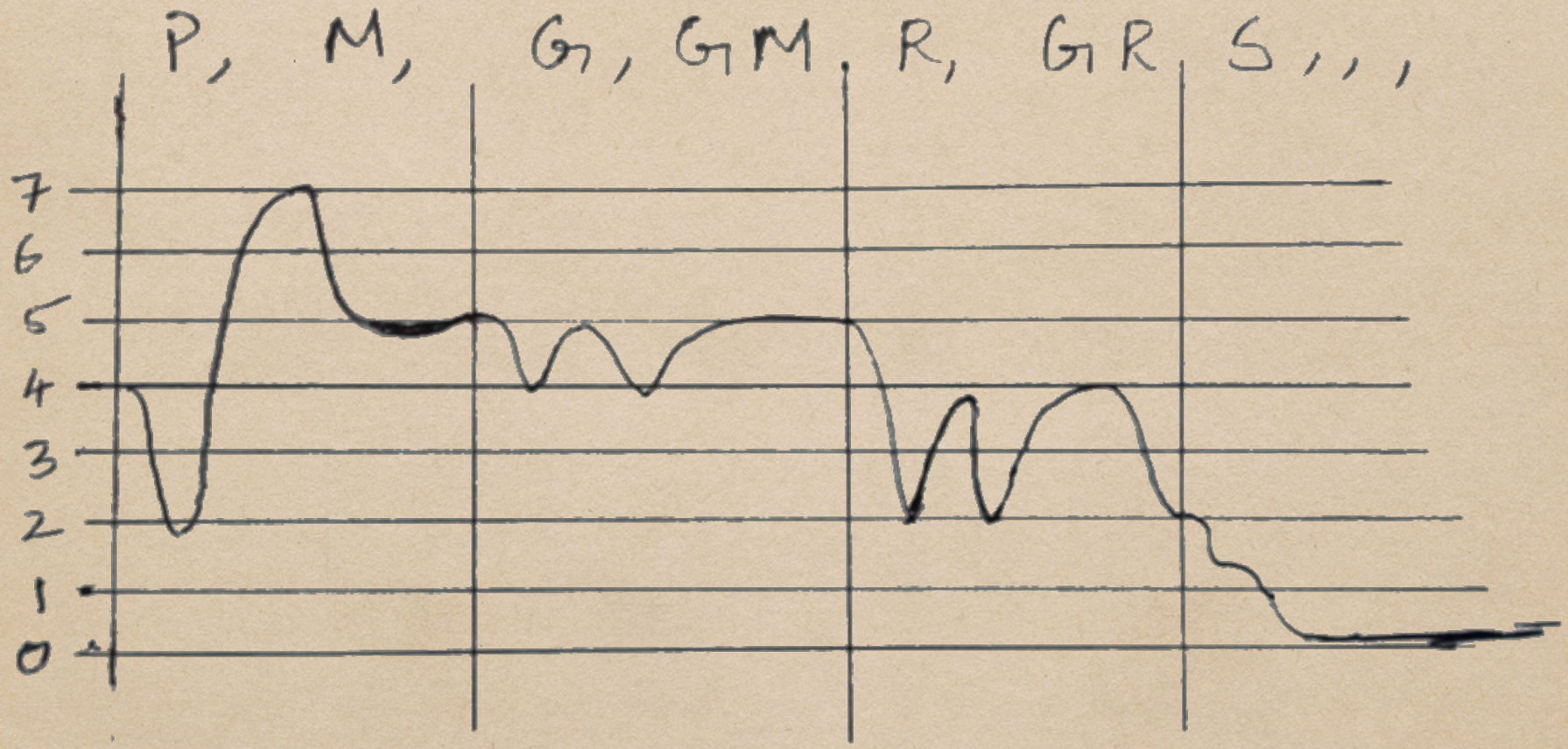
Me



Machine



Prescriptive vs Descriptive



Note uttered vs Note sung

Gamakams

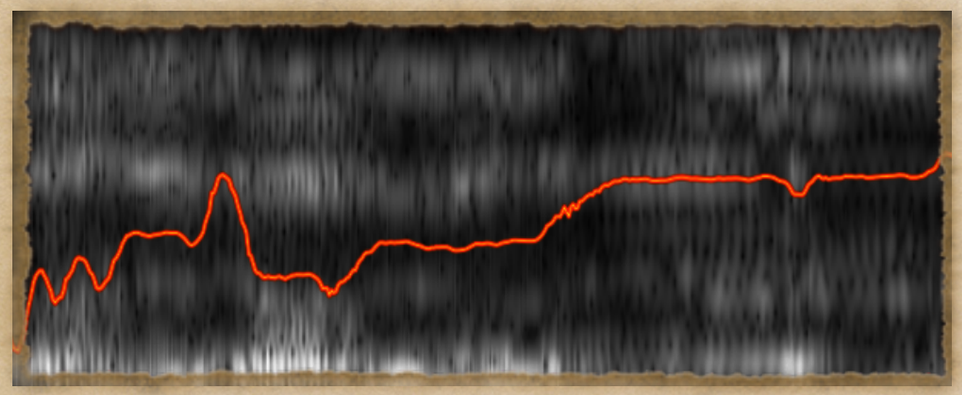
Modeling, Rendering

Gamakams

- Microtones, Continuous pitch movements
- Vocal Tradition
- Sangīta Sampradāya Pradarśini [Diskshitar 1904]

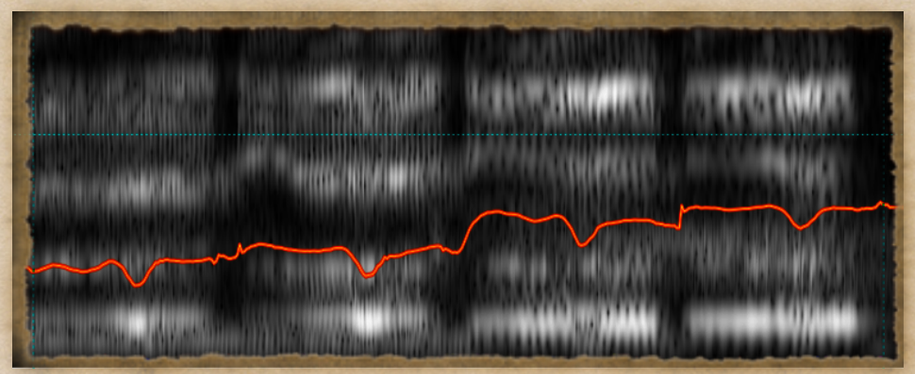
Gamakams

jāru



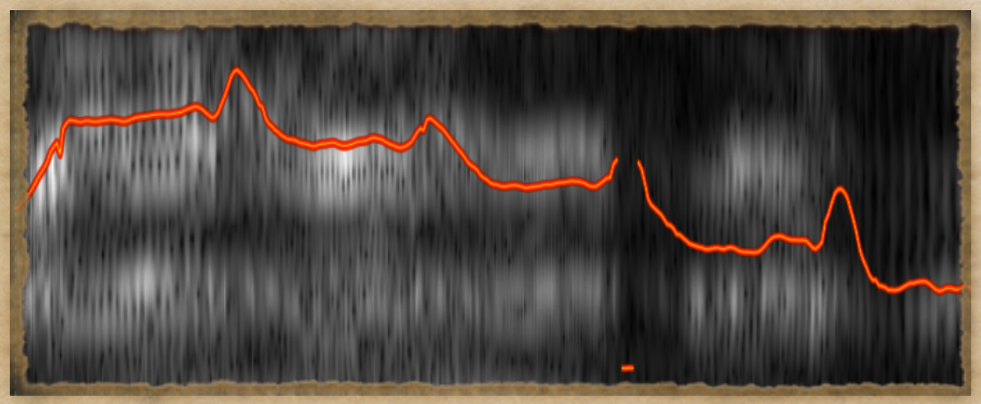
Upwards or downwards slide

sphuritam



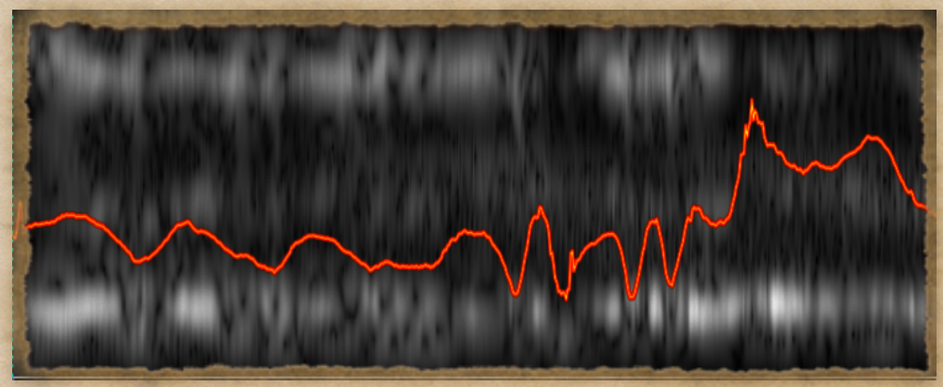
Emphasis on the second note

orikai



Downward slide with a twist

kampitam



Oscillations between two freqs

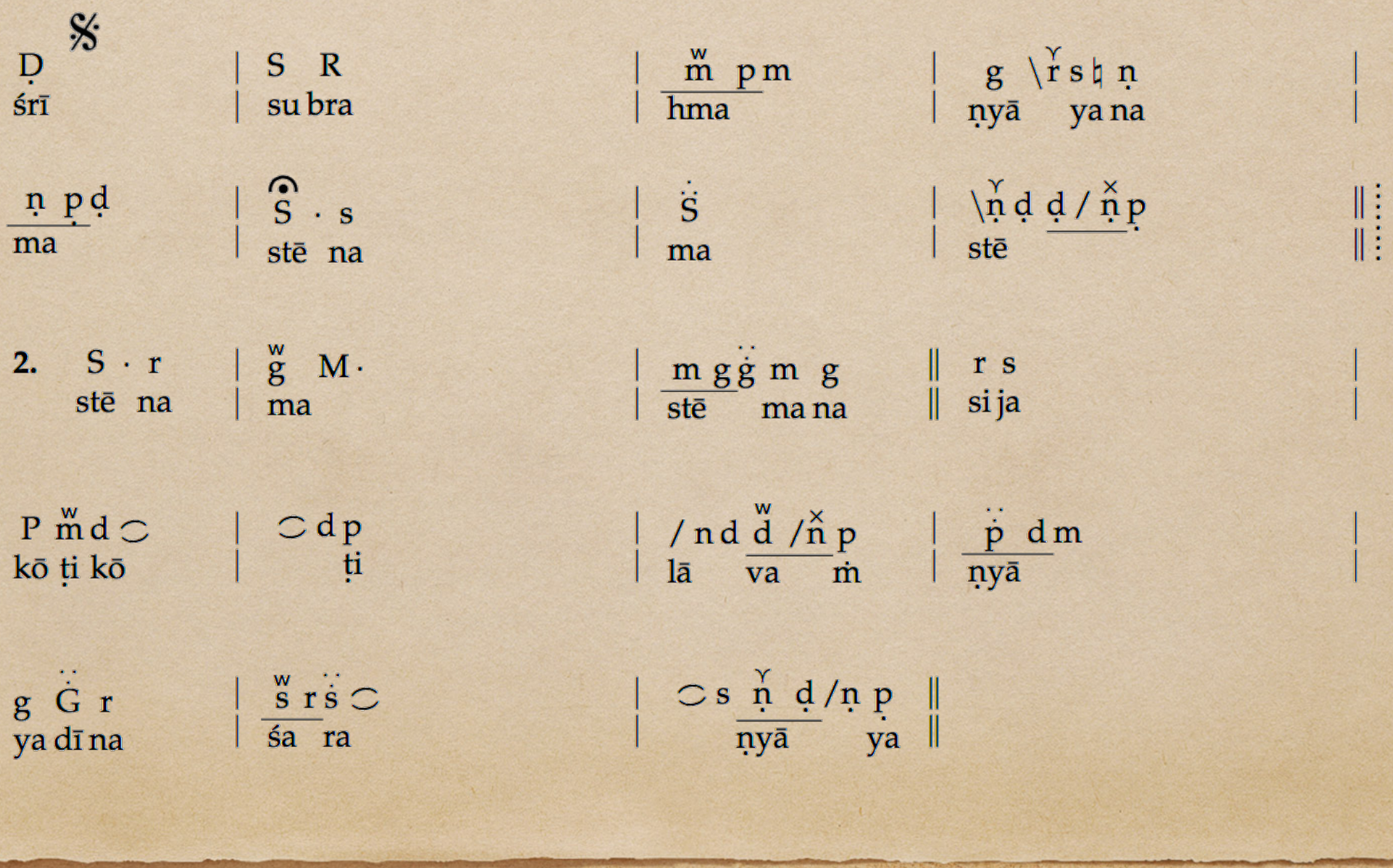
Gamakams in SSP

gamaka name	symbol	usage
kaṁpitam	~~~	G
sphuritam	∴	ṁ
pratyāhatam	∵	ṁ
nokku	w	g
Ravai	^	d
kaṇḍippu	✓	p
vaḷi	⁀	m
ētrajāru	/	/ g
iRakkajāru	\	\d
odukkal	×	n
orikai	ϒ	m
miśra gamakam		ṙ, g, p , etc.,

Gamakams in SSP

2. kīrtana— kāmbhōji rāga —rūpaka tāḷa

pallavi



Previous work

- Gaayaka [Subramanian, 2009]
  - database of phrases
  - "automatic gamakam" feature – guided
- PASR, DPASR transcriptions [Srikumar 2013]
  - Pitch, Attack, Sustain, Release
  - Phrases are split into 'Stage', and 'Dance' parts

Gaayaka

|S, N D|N S R G|

((P S,,)) , ((S , S>>> S)) -((D. S. D)) ((S , S>> S))- S R ((G<< G , ,))

- '<' and '>' increase and decrease pitch
- '(' and ')' are speed factors; deeper is faster

Overtone Gamakams

```
(defmacro g-inst [g-name g-env]
  `(definst ~g-name [~'f 260 ~'lf 260 ~'pf 260 ~'nf 260
                     ~'a 1 ~'s 1 ~'r 1
                     ~'d 1 ~'dp5 0.05 ~'d1 0.1 ~'d2 0.2 ~'d3 0.3
                     ~'d4 0.4 ~'d5 0.5 ~'d9 0.9]
     (let [gamakam-env# ~g-env]
       (~'* (sin-osc (env-gen gamakam-env#))
            (env-gen (envelope [0 1 1 0] [~'dp5 ~'d9 ~'dp5]))))))

(g-inst sphuritam-inst
        (envelope [f  f  lf  f  f]
                  [ d4 d1  d1 d4]
                  :welch))
```

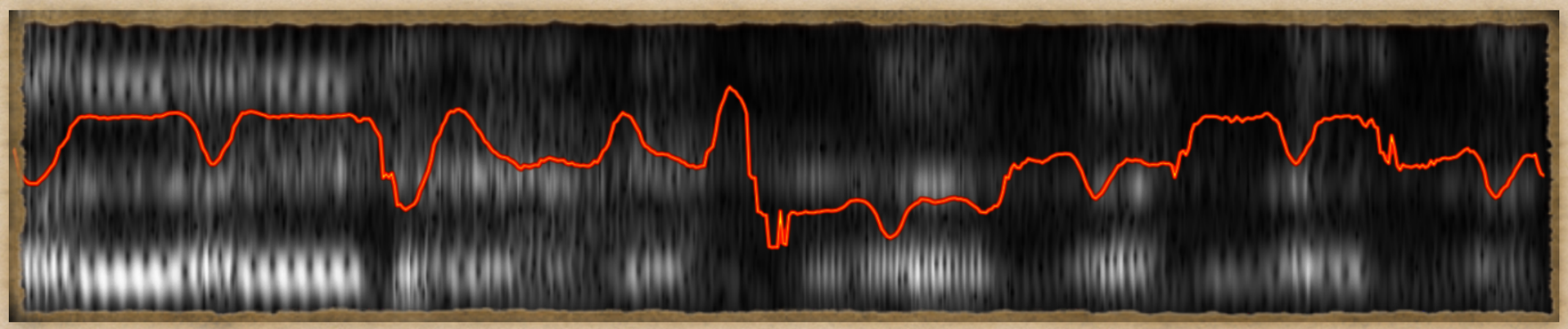
Overtone Gamakams

```
(g-inst sphuritam-inst
        (envelope [f  f  lf  f  f]
                  [ d4 d1  d1 d4]
                  :welch))

(g-inst jaru-inst
        (envelope [pf f  f]
                  [d9 d1]
                  :welch))

(g-inst kampitam-inst
        (envelope [pf f pf f]
                  [d5 d1 d4]
                  :welch))

(g-inst pasr-inst
        (envelope [pf f f nf]
                  [ a s r]
                  :welch))
```

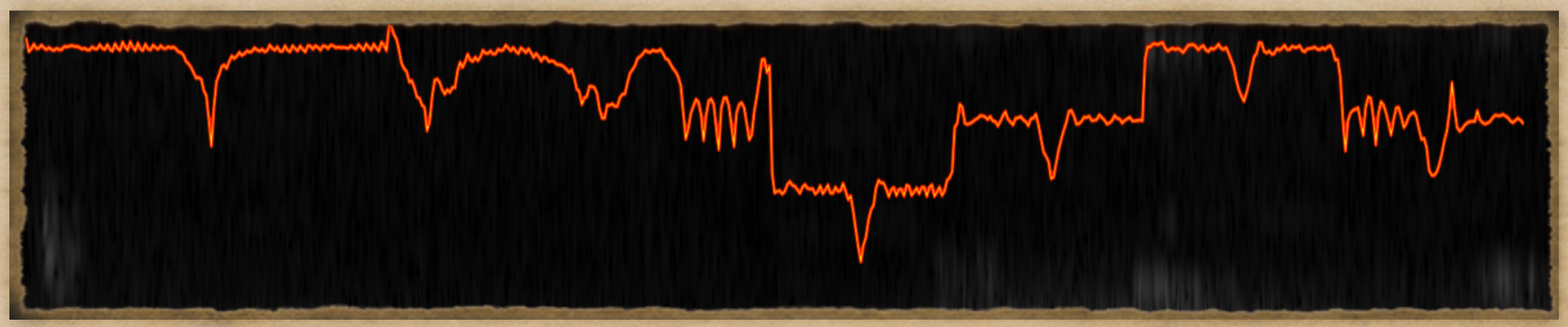
back to this...

```
(play-phrase
 (string->phrase (:mohana r/ragams)
                 "^g, ~r, ^s ^r ^g ^r"
                 1))
```

Me



Machine



Overtone Gamakams

```
(def gamakams-dict
  {:plain      {:inst plain-inst
                :inputs [:d :f]}
   :jaru       {:inst jaru-inst
                :inputs [:d1 :d9 :f :pf]}
   :sphuritam  {:inst sphuritam-inst
                :inputs [:f :lf :d1 :d4]}
   :kampitam   {:inst kampitam-inst
                :inputs [:f :pf :d1 :d4 :d5]}
   :kampitam-2 {:inst kampitam-inst-2
                :inputs [:f :pf :d1 :d2 :d3]}
   :pasr       {:inst pasr-inst
                :inputs [:f :pf :nf :a :s :r]}})

(defn with-synth-args [pre cur nex seconds a s r gamakam]
  (let [d seconds
        params {:pf (midi->hz pre)
                :f  (midi->hz cur)
                :nf (midi->hz nex)
                :lf (midi->hz (- cur 2))
                :a  a :s  s :r  r
                :d  seconds :dp5 (* 0.05 d) :d1 (* 0.1 d) :d2 (* 0.2 d)
                :d3 (* 0.3 d) :d4 (* 0.4 d) :d5 (* 0.5 d) :d9 (* 0.9 d)}
        {:keys [inst inputs]} (get gamakams-dict gamakam)
        args (flatten (vec (select-keys params (conj inputs :d9 :d1 :dp5))))]
    (apply inst args)))
```

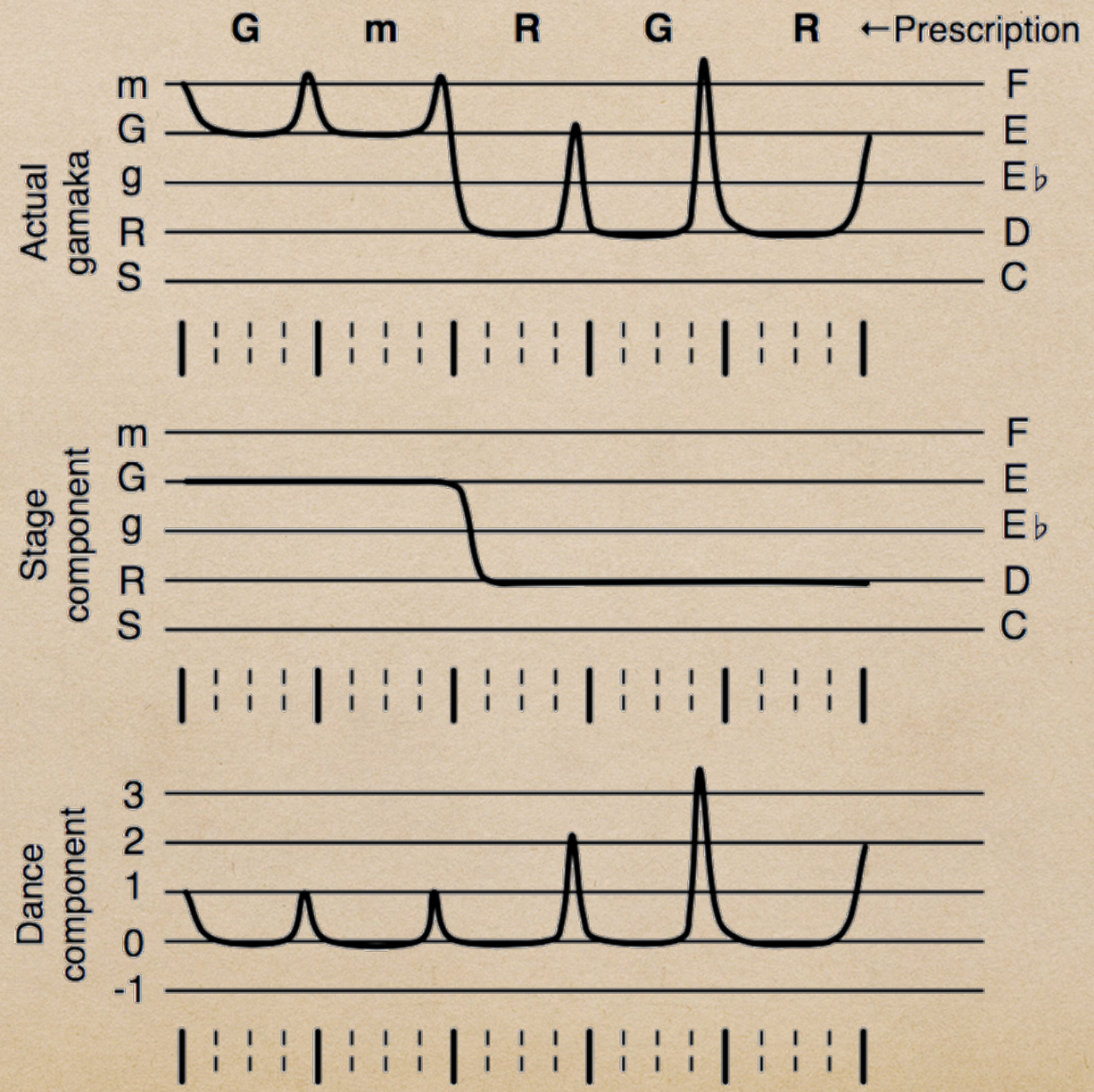
PASR

P, M, | G, GM | R, GR | S,,,

```
(["^pa:2"  [[4 0 0 0] [2 2 0 0] [7 2 4 0]]]
 ["^ma1:2" [[5 0 8 0]]]
 ["^ga3:2" [[5 0 6 0] [4 0.5 0 0.5] [5 1 0 0]]]
 ["^ga3"   [[5 0 0 0.5] [4 1 1 1] [5 0.5 0 0]]]
 ["ma1"    [[5 0 0 0.5] [4 1 1 1] [5 0.5 0 0]]]
 ["^ri2:2" [[5 0 0 0.5] [4 1 1 1] [5 0.5 0 0.5]
            [2 1 1 1] [4 0.5 0 0]]]
 ["^ga3"   [[4 0 0 0.5] [2 1 1 1] [5 0.5 0 0]]]
 ["ri2"    [[5 0 0 0.5] [2 1 1 1] [4 0.5 0 0]]]
 ["^sa:4"  [[0 0 16 0]]])
```

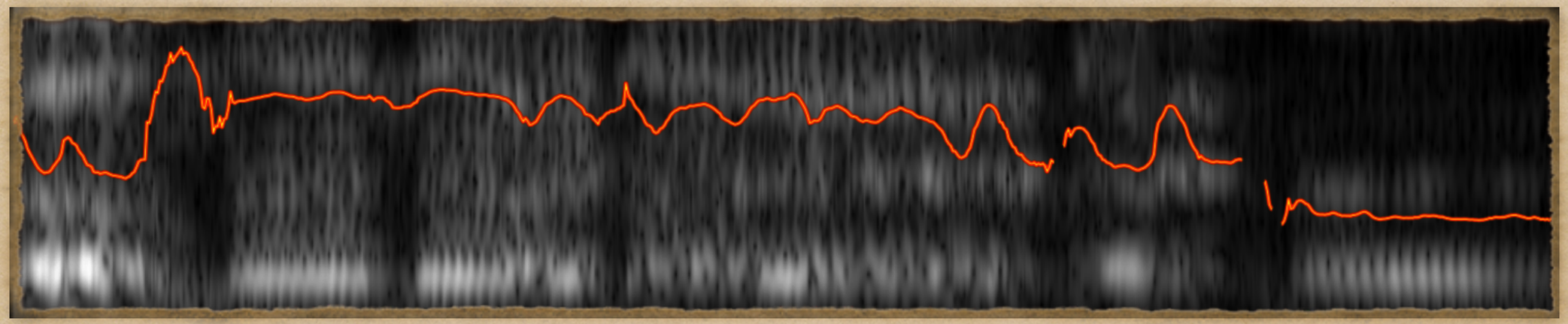
- durations, and pitch class in one part
- Vector specifies the PASR vars for each prescriptive note

DPASR

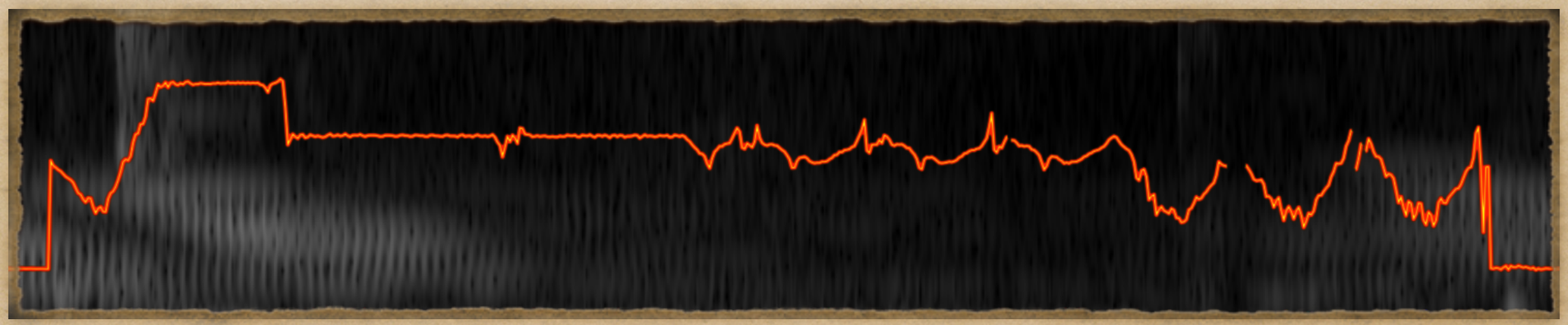

G m R G R ←Prescription
Actual gamaka
m G ɡ R S
F E E♭ D C
Stage component
m G ɡ R S
F E E♭ D C
Dance component
3 2 1 0 -1

Rendering PASR

Me



Machine



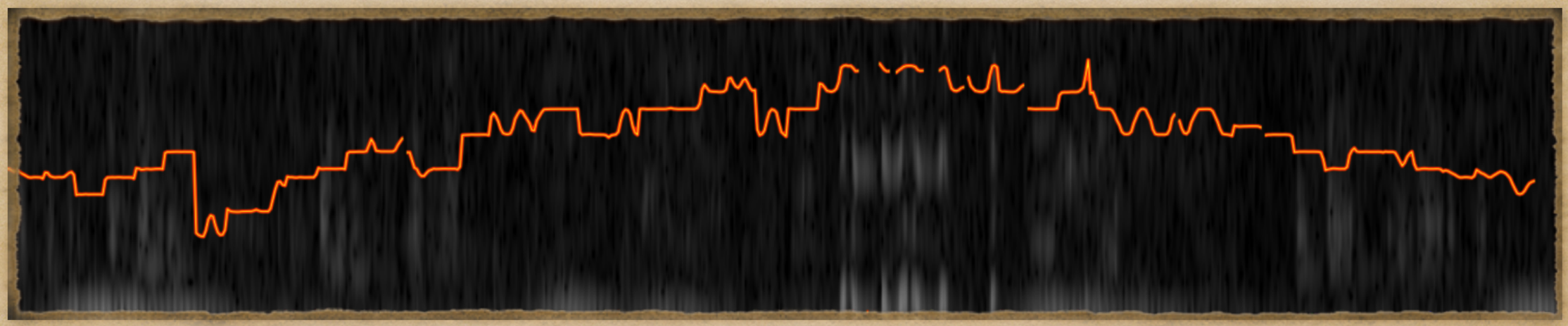
Rendering PASR

D n D -D P- P D P P m- D P m- G m R ||

G m P- ṇ S R G m | P , m , D n Ṡ , ||

D n Ṡ Ṙ , Ṙ- n Ṡ Ṙ Ġ ṁ Ṙ , Ġ Ṙ Ṡ ||

Ṙ n ,- Ṡ D ,- n D | P- D P P m- G m R ||



The ways of the future

As I see it

The path I see

- Have the basic abstractions, fair models to render
- Understanding the abstractions in melody
- Model rules in the ragams, generate music weighted by probabilities given by these rules
- Deep learning, Recurrent Neural Networks
- Improved transcription of Carnatic Music
- Open Carnatic Music Database
- (-> OCMD
  auto-transcription
  ML
  machine-created-music)

Is that music though?

Maybe?

References

- My work so far
  - https://github.com/ssrihari/ragavardhini
- SSP - 1905 - Sangita Sampradaya Pradarshini
  - http://ibiblio.org/guruguha/ssp_cakram1-4.pdf
- SPS - 2013 - Modeling Gamakas of Carnatic Music as a Synthesizer for Sparse Prescriptive Notation
  - http://sriku.org/thesis/srikumar-phd-thesis-cnm-modeling-gamakas-6aug2013.pdf
  - http://sitardivin.globat.com/seminar2013/047SrikumarS.pdf
- TMK - 2012 - Svara, Gamaka, Phraseology and Raga identity
  - http://compmusic.upf.edu/system/files/static_files/03-T-M-Krishna-2nd-CompMusic-Workshop-2012_0.pdf
- Gayaka - the software. Runs on windows, written in javascript by SPS.
  - http://carnatic2000.tripod.com/gaayaka6.htm
- SPS A Composition Based Method for Modeling Carnatic Music Rāgas and Style
  - http://sitardivin.globat.com/seminar2013/047SrikumarS.pdf
- Recent work on digitizing some carnatic music notation
  - http://devagitam.in

Carnatic Music Synthesis

in Clojure

@SrihariSriraman

@nilenso