Heuristic Pitch Finding for Square Note Notation

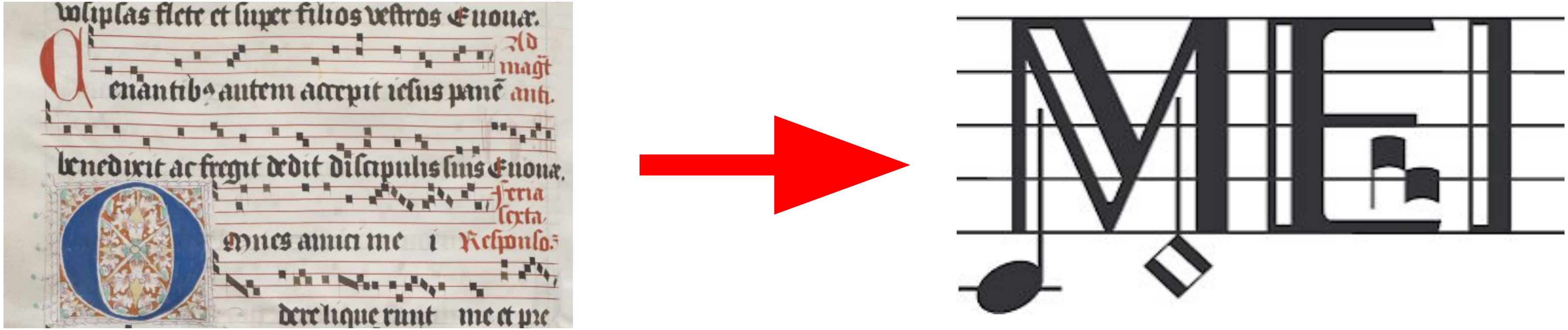
Noah Baxter

SIMSSA Workshop XIV
May 28th, 2018

DDMAL

Heuristic Pitch Finding

- Follows document analysis and classification stage
- Heuristically finds the pitch of each *glyph* in an image based on its position relative to discovered staff and clef positions
- Uses calculated pitches and original connected components to generate *symbolic notation*

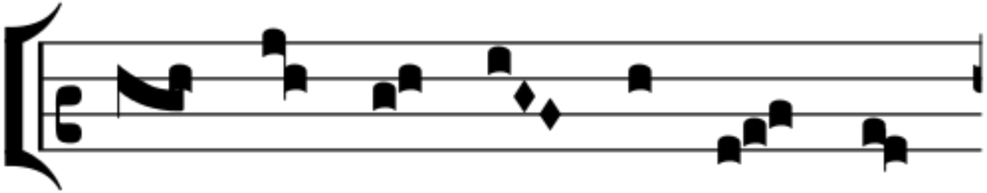


MEI - Music Encoding Initiative

“A system for encoding musical documents in a machine-readable structure”

- Created by Perry Roland in 1999
- Can be easily engraved (rendered) using tools like Verovio

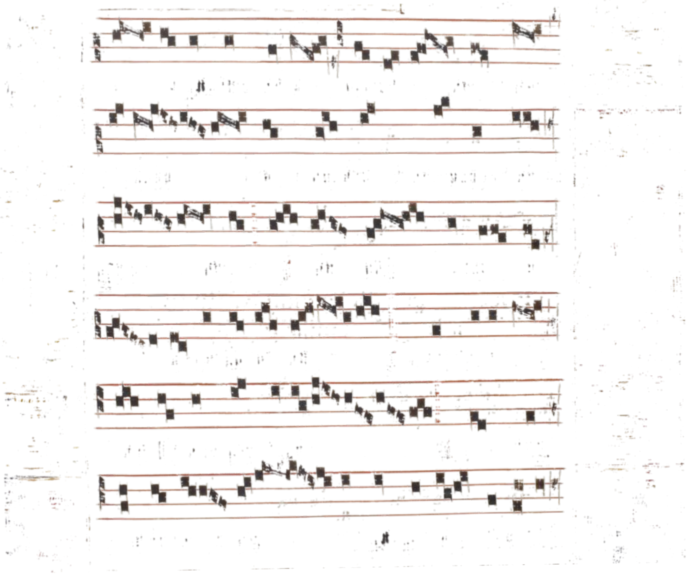
```
<staff n="1" label="feature-example">
<layer>
<syllable>
          <syl n= "initial">
          <rend color= "red"> O </rend>
          </syl>
          <!-- porrectus -->
          <neume>
          <nc oct= "3" pname= "e"/>
          <nc intm="d" oct= "3" pname= "d"/>
          <nc intm="u" oct= "3" pname= "e"/>
          </neume>
</syllable>
<syllable>
```



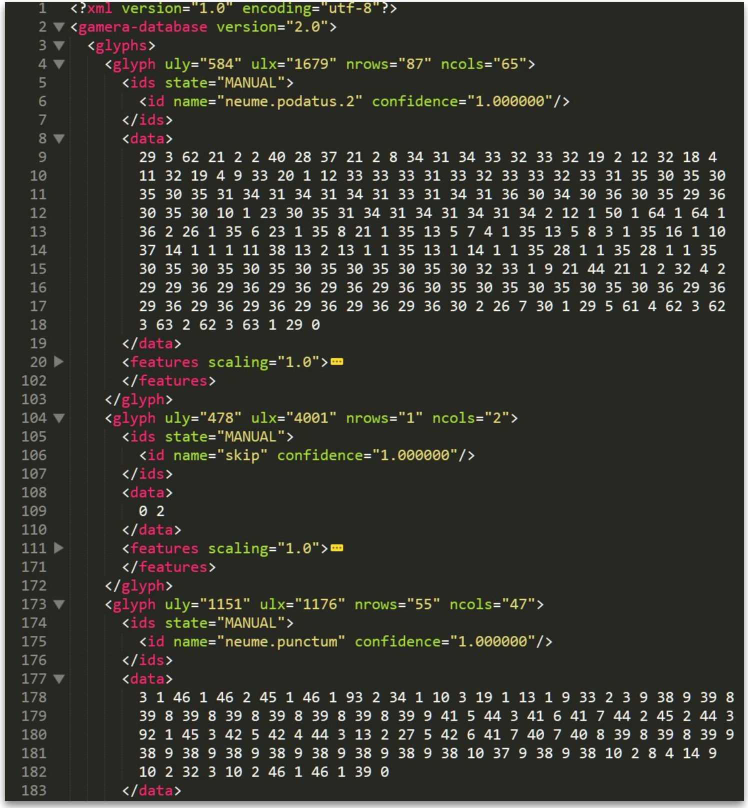
http://music-encoding.org/about/

What do we start with?

An image of stafflines and glyphs



GameraXML of classified connected components (CC)



```xml
<?xml version="1.0" encoding="utf-8"?>
<gamera-database version="2.0">
  <glyphs>
    <glyph uly="584" ulx="1679" nrows="87" ncols="65">
      <ids state="MANUAL">
        <id name="neume.podatus.2" confidence="1.000000"/>
      </ids>
      <data>
        29 3 62 21 2 2 40 28 37 21 2 8 34 31 34 33 32 33 32 19 2 12 32 18 4
        11 32 19 4 9 33 20 1 12 33 33 33 31 33 32 33 33 32 33 31 35 30 35 30
        35 30 35 31 34 31 34 31 34 31 33 31 34 31 36 30 34 30 36 30 35 29 36
        30 35 30 10 1 23 30 35 31 34 31 34 31 34 31 34 2 12 1 50 1 64 1 64 1
        36 2 26 1 35 6 23 1 35 8 21 1 35 13 5 7 4 1 35 13 5 8 3 1 35 16 1 10
        37 14 1 1 1 11 38 13 2 13 1 1 35 13 1 14 1 1 35 28 1 1 35 28 1 1 35
        30 35 30 35 30 35 30 35 30 35 30 35 30 32 33 1 9 21 44 21 1 2 32 4 2
        29 29 36 29 36 29 36 29 36 29 36 30 35 30 35 30 35 30 35 30 36 29 36
        29 36 29 36 29 36 29 36 29 36 29 36 30 2 26 7 30 1 29 5 61 4 62 3 62
        3 63 2 62 3 63 1 29 0
      </data>
      <features scaling="1.0">…
      </features>
    </glyph>
    <glyph uly="478" ulx="4001" nrows="1" ncols="2">
      <ids state="MANUAL">
        <id name="skip" confidence="1.000000"/>
      </ids>
      <data>
        0 2
      </data>
      <features scaling="1.0">…
      </features>
    </glyph>
    <glyph uly="1151" ulx="1176" nrows="55" ncols="47">
      <ids state="MANUAL">
        <id name="neume.punctum" confidence="1.000000"/>
      </ids>
      <data>
        3 1 46 1 46 2 45 1 46 1 93 2 34 1 10 3 19 1 13 1 9 33 2 3 9 38 9 39 8
        39 8 39 8 39 8 39 8 39 8 39 9 41 5 44 3 41 6 41 7 44 2 45 2 44 3
        92 1 45 3 42 5 42 4 44 3 13 2 27 5 42 6 41 7 40 7 40 8 39 8 39 8 39 9
        38 9 38 9 38 9 38 9 38 9 38 9 38 9 38 10 37 9 38 9 38 10 2 8 4 14 9
        10 2 32 3 10 2 46 1 46 1 39 0
      </data>
```

What is GameraXML?

What is GameraXML?

But first, what IS Gamera?

Gamera - The Kaiju

- Resembles a giant turtle
- Able to walk on 4 or 2 legs at will
- Can breathes fire

- Doesn’t like the cold brrrr…

A brief intro to Gamera - The Software

"A framework for building document analysis applications"

- Allows for the development of extensions (toolkits) based on its core functionality and GUI using wxPython (...)

- Started by Michael Droettboom, Karl Mac Millan, and Ichiro Fujinaga at John Hopkins in 2001

- Gamera is inside Rodan!
    - Musicstaves is a Gamera toolkit used throughout the SIMSSA project

http://gamera.informatik.hsnr.de/

GameraXML

- XML format for storing 1-bit images, primarily as training data for a classifier
- Basic structure:

```
<?xml version="1.0" encoding="utf-8"?>
<gamera-database version="2.0">
        <glyphs>
                <glyph uly="242" ulx="1758" nrows="26" ncols="18">
                        <ids state="AUTOMATIC">
                                <id name="number.three" confidence="1.000000"/>
                        </ids>
                        <data>
                        6 4 12 9 8 12 5 6 1 7 4 5 3 7 2 6 4 6 2 7 3 6 2 7 3 6 3 6 3 6 3 4 5 6
                        4 3 4 7 5 3 2 6 7 11 7 11 8 11 6 5 1 7 2 6 4 6 2 6 4 15 3 14 4 14 4 6
                        2 6 3 7 2 15 4 12 7 10 11 3 9 0
                        </data>
                </glyph>
        </glyphs>
</gamera-database>
```

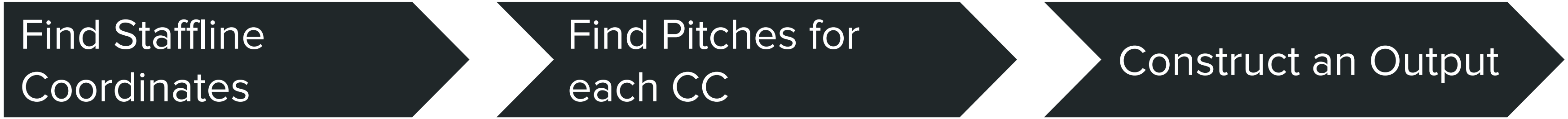
GameraXML

- Each glyph IS a *connected component* (pixel group)
- Based on glyph name and bounding box, center position can be determined (more on this later)

- Relatively easy to save/retrieve because GameraXML is just extended XML
- Contains all information needed to store all musical symbols as symbolic document notation

… except for pitch!

The 3-step pitch finding process



Find Staffline Coordinates

1. **Find bounding box of each staff on a page**
2. **Get many staffline points in each staff**

Find Pitches for each CC

1. **From each glyph, find its *center of mass***
2. **Compare against closest staffline points to find position. Compare against clef to find pitch**

Construct an Output

Generate MEI file from all CC + pitch pairs

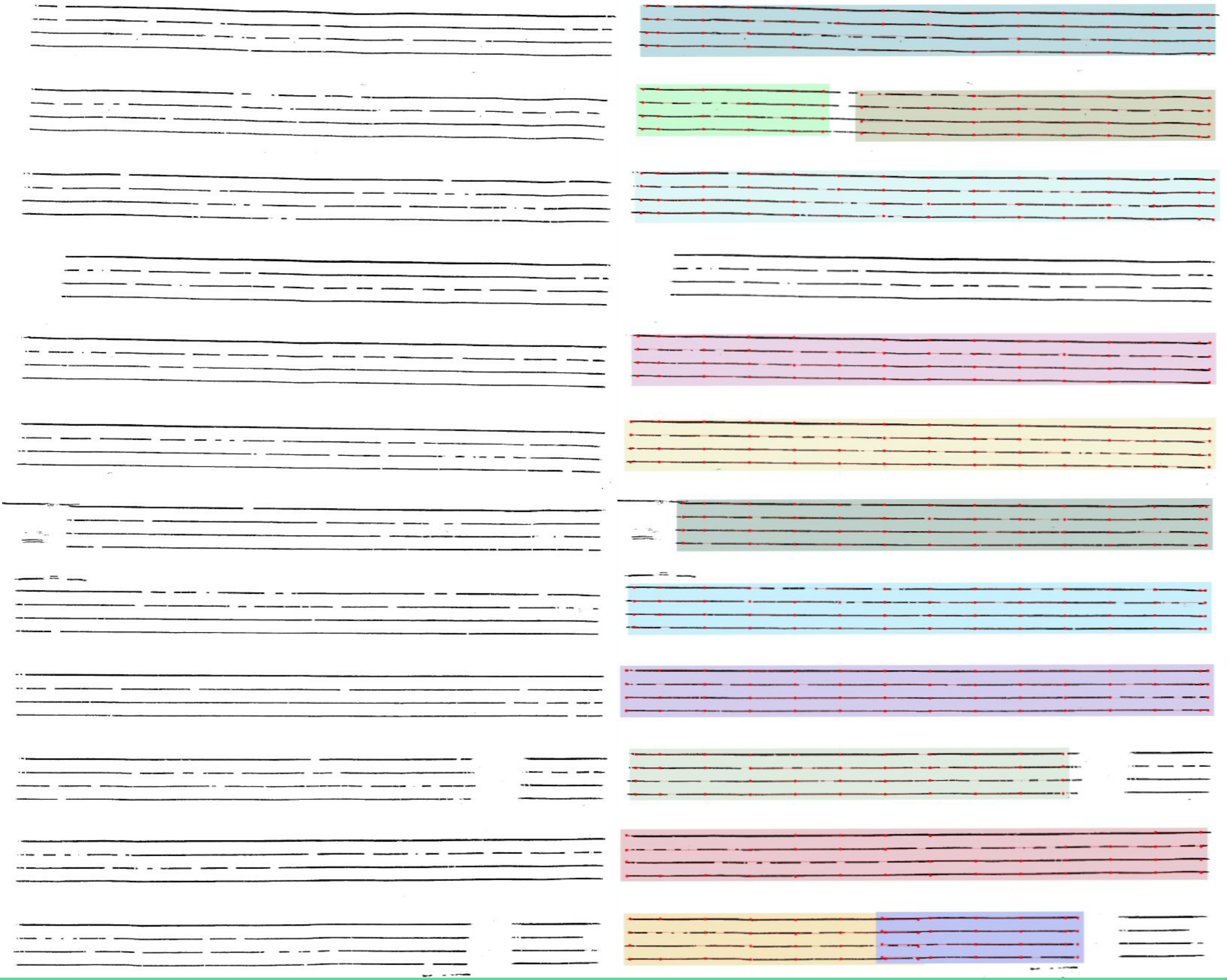
aOMR - *adaptive Optical Music Recognition*

- Started by Andrew Hankinson and Gabriel Vigliensoni at DDMAL in 2011
- A Gamera toolkit for converting an image + classified CC into MEI 1.0
  - Uses Musicstaves toolkit to find and remove stafflines from image
  - Calculates pitches
  - Using information given from CC file, generates MEI 1.0 output

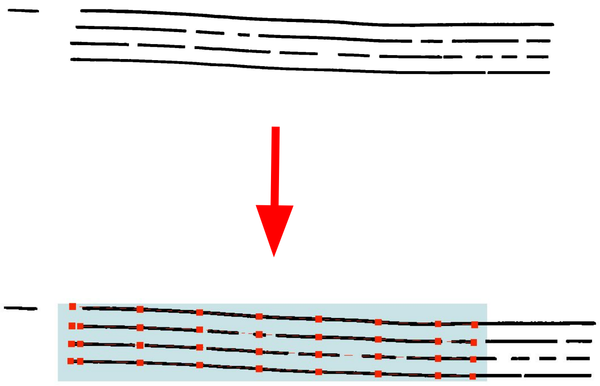
- Miyao staff finding algorithm
  - Algorithm written by Hidetoshi Miyao at Shinshu University in 2002, implemented with Musicstaves by Christoph Dalitz
  - For each staff, finds points along x-axis slices each line
  - Works on non-straight and non-perfectly horizontal lines!

Intelligent Data Engineering and Automated Learning —
IDEAL 2002: Third International Conference Manchester, UK,
August 12–14, 2002 Proceedings (pp.562-568)

https://github.com/DDMAL/aOMR/

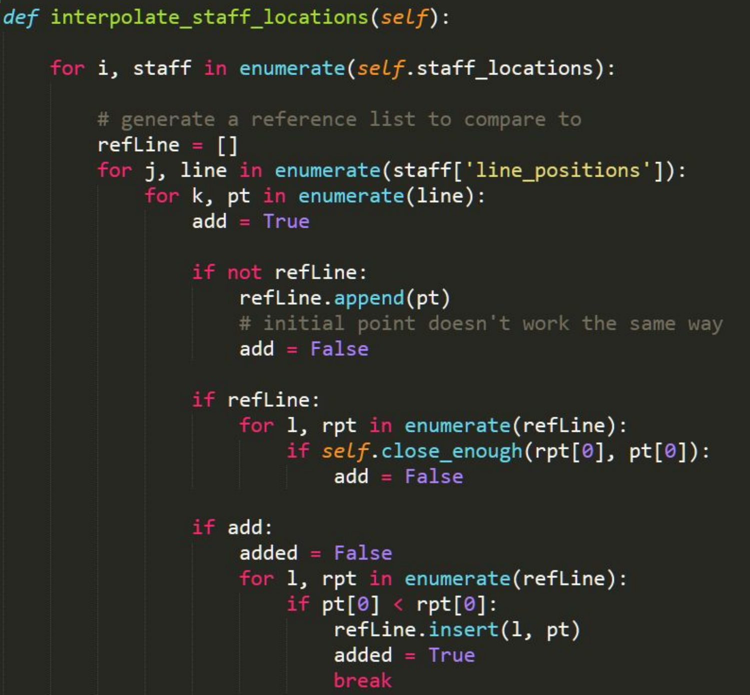


What does Musicstaves see?

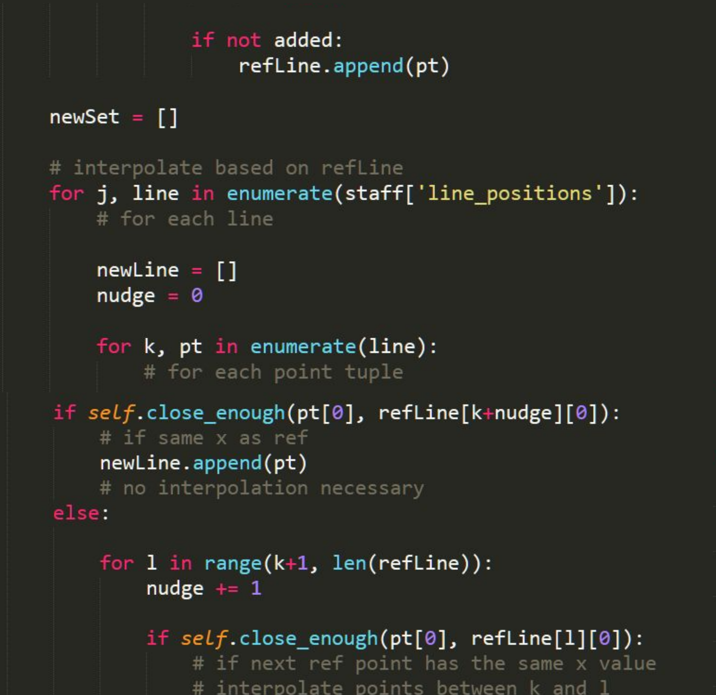


aOMR - *adaptive Optical Music Recognition*

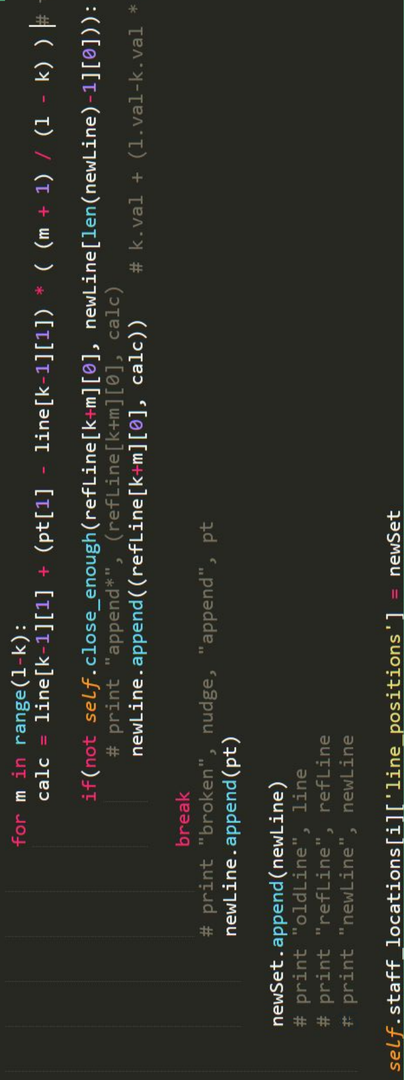
- Line point interpolation (a thing I did!)



```
def interpolate_staff_locations(self):

    for i, staff in enumerate(self.staff_locations):

        # generate a reference list to compare to
        refLine = []
        for j, line in enumerate(staff['line_positions']):
            for k, pt in enumerate(line):
                add = True

                if not refLine:
                    refLine.append(pt)
                    # initial point doesn't work the same way
                    add = False

                if refLine:
                    for l, rpt in enumerate(refLine):
                        if self.close_enough(rpt[0], pt[0]):
                            add = False

                if add:
                    added = False
                    for l, rpt in enumerate(refLine):
                        if pt[0] < rpt[0]:
                            refLine.insert(l, pt)
                            added = True
                            break
```



```
            if not added:
                refLine.append(pt)

newSet = []

# interpolate based on refLine
for j, line in enumerate(staff['line_positions']):
    # for each line

    newLine = []
    nudge = 0

    for k, pt in enumerate(line):
        # for each point tuple

if self.close_enough(pt[0], refLine[k+nudge][0]):
    # if same x as ref
    newLine.append(pt)
    # no interpolation necessary
else:

    for l in range(k+1, len(refLine)):
        nudge += 1

        if self.close_enough(pt[0], refLine[l][0]):
            # if next ref point has the same x value
            # interpolate points between k and l
```



```
                    for m in range(l-k):
                        calc = line[k-1][1] + (pt[1] - line[k-1][1]) * ( (m + 1) / (l - k) ) #

                        if(not self.close_enough(refLine[k+m][0], newLine[len(newLine)-1][0])):
                            # print "append*", (refLine[k+m][0], calc)
                            newLine.append((refLine[k+m][0], calc))    # k.val + (l.val-k.val *

                    break
            # print "broken", nudge, "append", pt
            newLine.append(pt)

    newSet.append(newLine)
    # print "oldLine", line
    # print "refLine", refLine
    # print "newLine", newLine

self.staff_locations[i]['line_positions'] = newSet
```

Interpolating non-perfect staff lines

- Here is an example of line output from the Miyao staff finder:

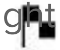
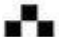
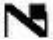
```
[[(1018, 674), (1050, 674), (1261, 679), (1471, 686), (1681, 701), (1891, 711), (2101,
718), (2312, 725), (2484, 731), (2481, 790), (2312, 785), (2101, 779), (1891, 772),
(1681, 761), (1471, 748), (1261, 740), (1050, 735), (1015, 734)]]
```

- Generates a reference line by making a point at each x position across all lines
- Parses each line in each Staff from the grouping of staves
  - For each point in the ref, if a point doesn’t exist at that x, generate it
  - Output completed lines for each staff line from the combination of preexisting and generated staffline points

The actual pitch finding part

For each glyph, find a center weighting

- Some glyph types are treated differently, not simply the center of bounding box
- For comple glyphs, weig t the first pitch lculate followi itches in relation

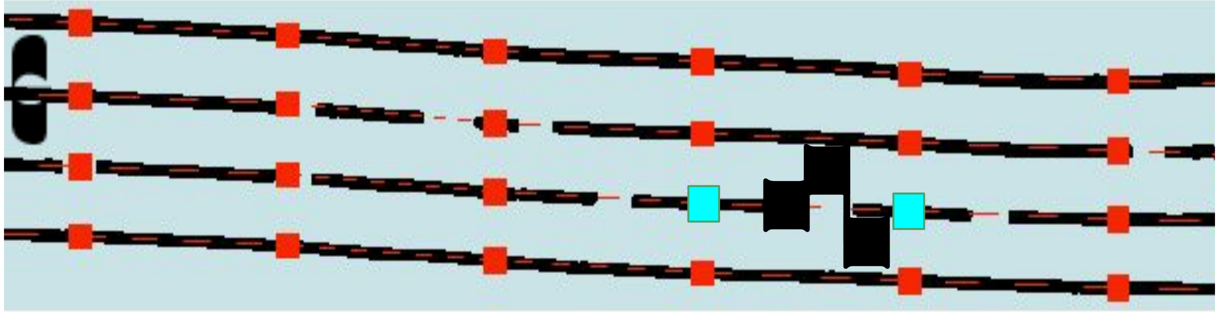




- Punctum Clivis Torculus Porrectus
- Intervallic differences determines pre/proceeding pitches
- Contours determine complex neume types (u u d u)

- If within a certain margin of the line between the two closest (x, y) points on either side, line, otherwise space
- The specific space/line determines the pitch (in relation to the clef)

A pitch finding example

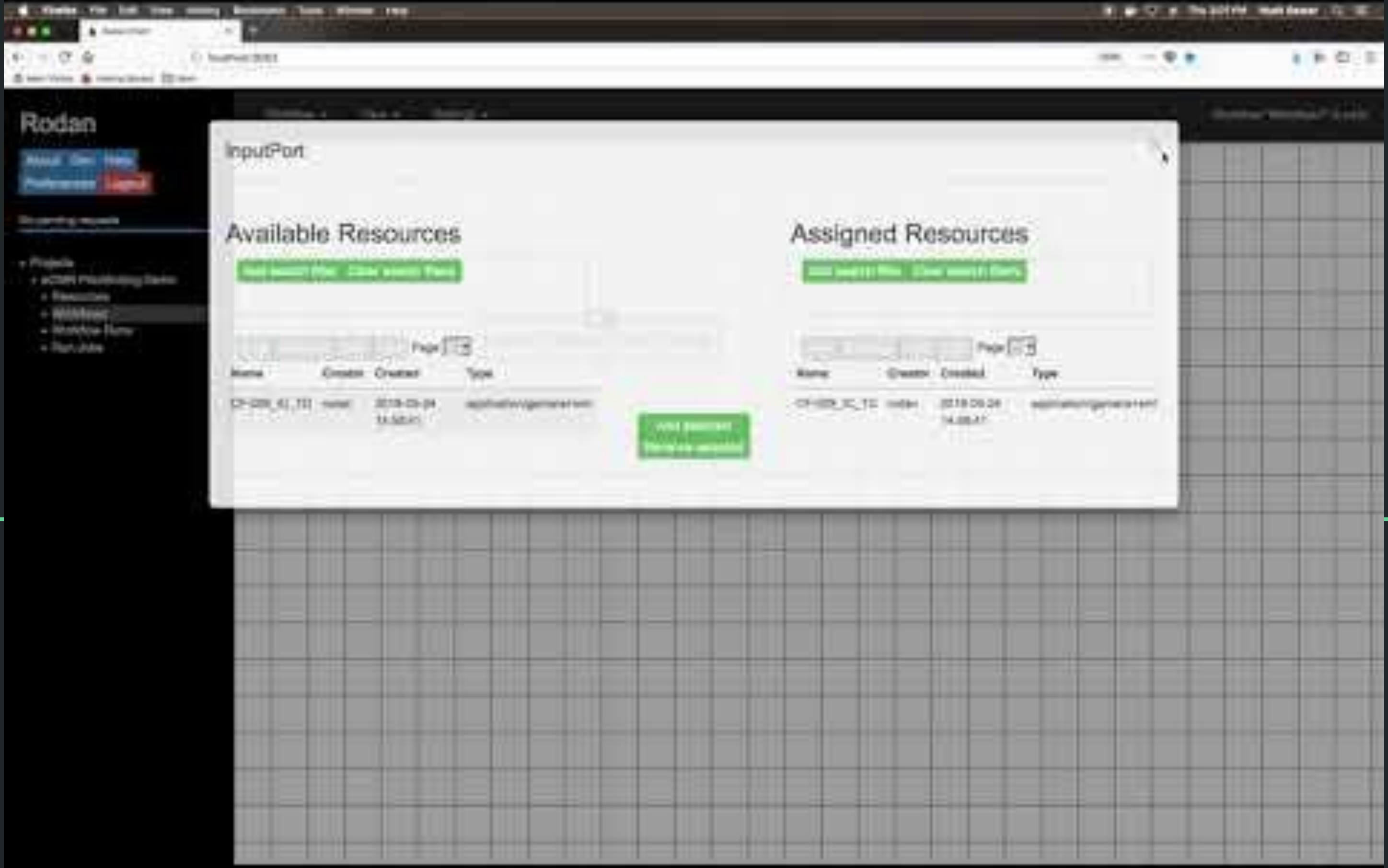
```
<glyph uly="242" ulx="1758" nrows="26" ncols="18">
        <ids state="MANUAL">
                <id name="neume.torculus.2.3" confidence="1.000000"/>
        </ids>
        <data>
                …
        </data>
        <features scaling="1.0">
                …
        </features>
</glyph>
```



Finally, convert the CCs and pitches into MEI

- Except MEI is changing
  - So instead of hardcoding this conversion, separate into 2 parts
    1. Image & CC into json object containing CC + Pitch info
    2. json into MEI X.X

  - When next MEI standard comes about, simpler process to update output



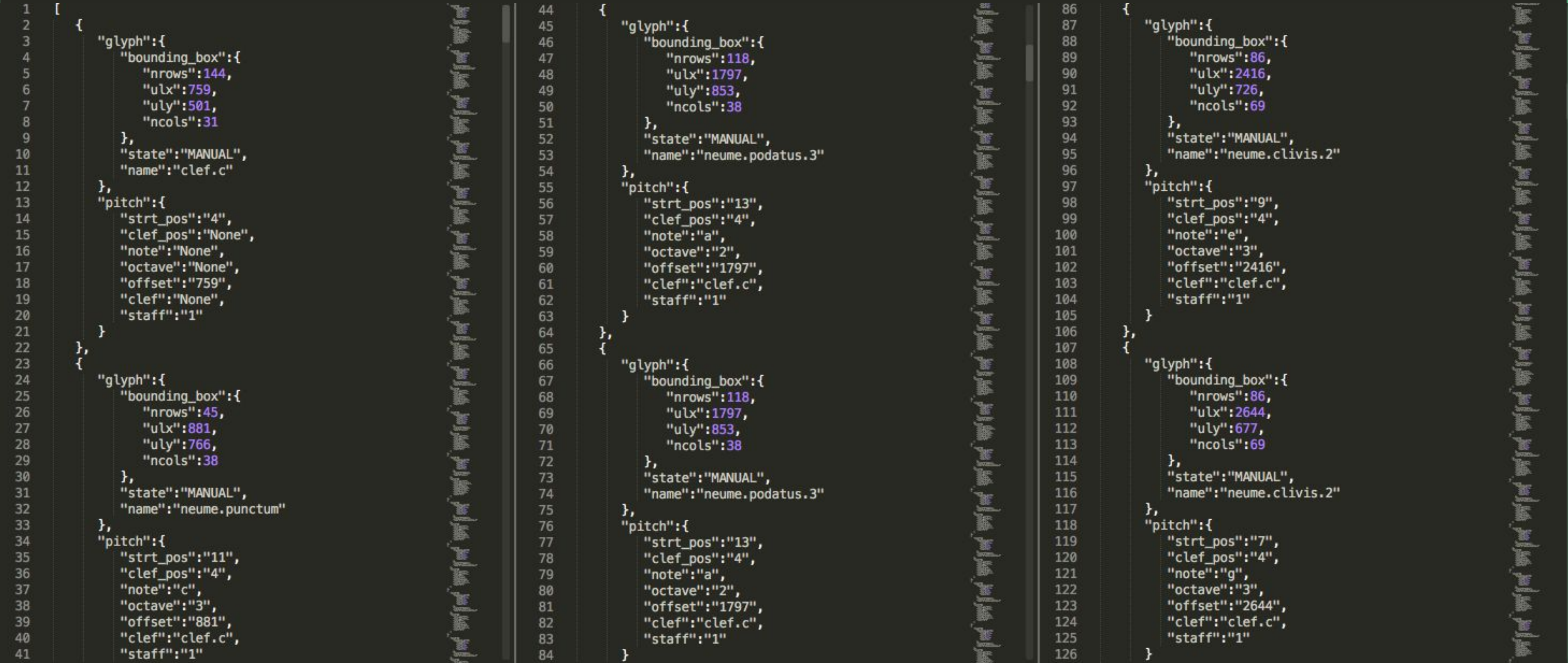
Available Resources

Name	Creator	Created	Type
CF-009_41_TD	rodan	2019-03-24 14:50:41	application/gamera+xml

Assigned Resources

Name	Creator	Created	Type
CF-009_3C_TC	rodan	2019-03-24 14:08:47	application/gamera+xml

What do we get in return?



```
[
    {
        "glyph":{
            "bounding_box":{
                "nrows":144,
                "ulx":759,
                "uly":501,
                "ncols":31
            },
            "state":"MANUAL",
            "name":"clef.c"
        },
        "pitch":{
            "strt_pos":"4",
            "clef_pos":"None",
            "note":"None",
            "octave":"None",
            "offset":"759",
            "clef":"None",
            "staff":"1"
        }
    },
    {
        "glyph":{
            "bounding_box":{
                "nrows":45,
                "ulx":881,
                "uly":766,
                "ncols":38
            },
            "state":"MANUAL",
            "name":"neume.punctum"
        },
        "pitch":{
            "strt_pos":"11",
            "clef_pos":"4",
            "note":"c",
            "octave":"3",
            "offset":"881",
            "clef":"clef.c",
            "staff":"1"
```

```
    {
        "glyph":{
            "bounding_box":{
                "nrows":118,
                "ulx":1797,
                "uly":853,
                "ncols":38
            },
            "state":"MANUAL",
            "name":"neume.podatus.3"
        },
        "pitch":{
            "strt_pos":"13",
            "clef_pos":"4",
            "note":"a",
            "octave":"2",
            "offset":"1797",
            "clef":"clef.c",
            "staff":"1"
        }
    },
    {
        "glyph":{
            "bounding_box":{
                "nrows":118,
                "ulx":1797,
                "uly":853,
                "ncols":38
            },
            "state":"MANUAL",
            "name":"neume.podatus.3"
        },
        "pitch":{
            "strt_pos":"13",
            "clef_pos":"4",
            "note":"a",
            "octave":"2",
            "offset":"1797",
            "clef":"clef.c",
            "staff":"1"
        }
```

```
    {
        "glyph":{
            "bounding_box":{
                "nrows":86,
                "ulx":2416,
                "uly":726,
                "ncols":69
            },
            "state":"MANUAL",
            "name":"neume.clivis.2"
        },
        "pitch":{
            "strt_pos":"9",
            "clef_pos":"4",
            "note":"e",
            "octave":"3",
            "offset":"2416",
            "clef":"clef.c",
            "staff":"1"
        }
    },
    {
        "glyph":{
            "bounding_box":{
                "nrows":86,
                "ulx":2644,
                "uly":677,
                "ncols":69
            },
            "state":"MANUAL",
            "name":"neume.clivis.2"
        },
        "pitch":{
            "strt_pos":"7",
            "clef_pos":"4",
            "note":"g",
            "octave":"3",
            "offset":"2644",
            "clef":"clef.c",
            "staff":"1"
        }
```

aOMR

- A Gamera toolkit
  - Can be used inside Gamera GUI
- Uses Miyao or Avg Line algorithm

aOMR-Miyao

- A Rodan job
  - Can be used within Rodan workflows
- Only uses Miyao algorithm

SIMSSA | Single Interface for Music Score Searching and Analysis

Verovio

MEI

Social Sciences and Humanities Research Council of Canada

Conseil de recherches en sciences humaines du Canada

Canada

McGill

Schulich School of Music
École de musique Schulich

DDMAL

DISTRIBUTED DIGITAL MUSIC ARCHIVES & LIBRARIES LAB

CIRMMT

Centre for Interdisciplinary Research in Music Media and Technology

Fonds de recherche Société et culture

Québec

Summer goals for pitch finding in Rodan

1. Implement aOMR into Rodan job wrapper
2. Create new json output for GameraXML CC + pitch information
3. Create rodan job for converting ^^ -> MEI 3.0/4.0/X.X
4. Investigate machine learning methods for pitch finding
5. Evaluate performance of each approach